IDA

INSTITUTE FOR DEFENSE ANALYSES

# Interpretation of the Department of Defense Goal Security Architecture Using the Reference Model for Open Distributed Processing


Edward A. Schneider, Task Leader

Edward A. Feustel



November 1999

Approved for public release; distribution unlimited.

IDA Paper P-3504

Log: H 99-002793

This work was conducted under contract DASW01 98 C 0067, Task DD-5-1671, for the National Security Agency. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

INSTITUTE FOR DEFENSE ANALYSES

# Interpretation of the Department of Defense Goal Security Architecture Using the Reference Model for Open Distributed Processing

Edward A. Schneider, Task Leader

Edward A. Feustel

## Preface

This paper was prepared by the Institute for Defense Analyses (IDA) for the National Security Agency under the task Research Plan for Information Domains. It responds to the task objective to systematically analyze and characterize information domains from each of the Reference Model for Open Distributed Processing viewpoints. The strengths and weaknesses of the information domain approach to information security will be examined from each viewpoint. Important research problems involving information domains that remain to be solved will be identified.

The following IDA research staff members were reviewers of this paper: Dr. Richard J. Ivanetich, Mr. Terry Mayfield, Dr. Reginald N. Meeson, and Dr. Richard P. Morton. Special thanks are due to Ms. Katydean Price for her comments and editing.

# Contents

## Contents

# Figures

## Tables

## Executive Summary

### Background

U.S. defense information systems and those of its coalition partners are increasingly networked among themselves and with other information systems, all of which are frequently managed by different organizations and subject to different security policies. This is especially true in coalition and joint operations where the networks may be very dynamic. These systems must be capable of protecting multiple changing classifications of information, thus expanding the requirements for MultiLevel Security. In previous research, the Institute for Defense Analyses found that the Department of Defense Goal Security Architecture (DGSA) provides a suitable model for expressing the information security of such systems.

### Results

This paper uses the Reference Model for Open Distributed Processing (RM-ODP), an international standard, to present the information system security concepts found in the DGSA. The result is several frameworks that can be instantiated to form an architecture for a particular system. Each framework represents a different concern of the security architect.

We begin with the definition and discussion of a constraint-based architecture. A constraint-based architecture is determined by an ordered set of prioritized constraints that reduce the number of implementations of the architecture as more constraints are added to the set. The ideal constraint-based architecture supplies no more constraints than absolutely required for the purpose of the architecture so that the architect/implementor has the greatest flexibility. In the event that the set of constraints permits more than one implementation, the constraint-based architecture can be said to be *abstract*. To focus on a specific function of the system, such as security, an abstract architecture is derived using an ordered subset of the constraints. This also preserves some properties of interest from the original architecture. This abstract architecture may be used to reason about the selected function.

We next describe the RM-ODP. The RM-ODP consists of five "viewpoints," that is, abstractions that focus on particular concerns within a distributed system or application description while leaving other concerns transparent (ignored in the viewpoint):

- Enterprise viewpoint: Purpose, scope, and policies for a system. Specifies the organizational security policies that must be enforced.
- Information viewpoint: Semantics of information and information processing. Specifies how security policies are represented, managed, and transformed.
- Computational viewpoint: Functional decomposition of a system into objects that interact at inter-

faces. Specifies the separation required for computations.

- Engineering viewpoint: Mechanisms and functions required to support distributed interaction between objects. Specifies the placement of security mechanisms in the "system of systems" to enforce the policy and the coordination that occurs between them.
- Technology viewpoint: Choice of technology used for the system. Specifies the standards, technologies, and components that may be used for security mechanisms.

Each viewpoint provides a selected set of architectural concepts and structuring rules that forms a descriptive language through which distributed systems and applications can be described. In this paper, we concentrate on the security aspects of each viewpoint.

We then describe the DGSA as a set of constraints within each viewpoint. It can therefore be thought of as a Constraint-Based Architecture family. Each of the constrained viewpoints becomes a framework that can be instantiated for the particular security needs for a system. This description of the DGSA provides a different perspective than previous formulations, and highlights the key principles that it embodies.

We present a simple military example that uses the military operations plan and order for a coalition task force employing forces drawn from three coalition nations. Relevant security requirements from the DGSA are imposed to aid the description of the system.

We conclude with a discussion of suggested future activity. One such activity is to investigate the isolation of DGSA information domains using commercially available virtual machines and virtual private networks. Another activity is to define an enterprise-level policy language that can be translated through the information and computational viewpoints to generate security service requirements in the engineering viewpoint

**Audience**

On finishing the paper, the reader should:

- understand the principles of the DGSA,
- have a rudimentary knowledge of the RM-ODP and understand why it is useful in describing security in distributed systems, and
- begin to understand how an automated system could be designed that would support coalition operations.

The paper is intended for a technical reader interested in how security can be provided within distributed systems and applications. The reader should have an extensive background in either technical security and/or distributed systems and applications.

# 1. Introduction

Defense information systems are increasingly networked to other information systems, which are frequently managed by different organizations and subject to different security policies. This is especially true in coalition and joint operations where the networks may be very dynamic. These systems must be capable of protecting multiple changing classifications of information, thus expanding the requirements for MultiLevel Security (MLS).

The Department of Defense (DoD) Goal Security Architecture (DGSA) [DGSA] has been in existence since 1995 as part of DoD's Technical Architecture Framework for Information Management (TAFIM). It describes the end point of an evolution of security architectures that both DoD and the civilian sector can use to model information security in a dynamic network. Unfortunately, the DGSA document and its successors have not successfully conveyed the meaning or the utility of the DGSA. Many who read the document expect an architecture to provide a map leading to an implementation. Instead, the DGSA provides a set of abstractions that forms a framework for discussing implementations and for determining how well these abstractions meet a series of DoD goals. Another major reason may be that discussion of such factors as information security policies, network configuration, security services, and security management are intermingled. In previous work [SFR97, S99], we have separated some of these factors by placing them on different axes of the design space.

The purpose of this paper is to use viewpoints from the Reference Model for Open Distributed Processing (RM-ODP) to improve the expression of the DGSA. In particular, we separate the description of various security services that are linked in the DGSA document [DGSA]. We also use DGSA information domains to relate RM-ODP viewpoints. As an example, we show the utility of the DGSA and the RM-ODP in modeling and understanding technical security for coalition forces at the system level. We assume a basic familiarity with security concerns on hosts and on networks, but not with DGSA or RM-ODP concepts.

The RM-ODP [PART1, PART2, PART3, PART4] is both an International Organization for Standardization standard (ISO/IEC 107460) and an International Telecommunication Union Recommendation (ITU-T X.901-X.904). In the same way that the ISO Reference Model for Open Systems Interconnection (OSI) provides a method for description that has transformed the way in which we describe and explain communication, the RM-ODP provides a method to

describe distributed systems and their features. In particular, the RM-ODP can be used to describe a distributed system that embodies all Quality of Service (QoS) parameters related to security, often called Qualities of Protection (QoP). These QoS parameters include the following:

- Confidentiality: Information is not made available or disclosed to unauthorized individuals, entities, or processes.
- Integrity: Code or data is not altered or destroyed in an unauthorized manner.
- Availability: Access to data and information services is timely and reliable for authorized users.
- Accountability: System activities can be traced to persons or processes that may then be held responsible.

The RM-ODP features five viewpoints of a system (or system of systems), as depicted in Table 1.

**Table 1. RM-ODP Viewpoints**

| Viewpoint | Description | Security Aspects Treated |
|---|---|---|
| Enterprise | The purpose for the system, the use of the system, and the policies for the use of and management of the system. | Organizational security policies specifying the QoP objectives. |
| Information | The information entities to be represented and manipulated in the operation of the enterprise. | Association between information entities and security policies. |
| Computational | The model for the transformation of information of the enterprise. | Isolation of computations. |
| Engineering | The model supporting the computational viewpoint that provides "distribution transparencies," including access, failure, location, migration, relocation, replication, persistence, and transaction transparencies. | Security services that satisfy the QoP objectives. |
| Technology | The model describing the components and standards used in the engineering viewpoint. | |

All viewpoints show aspects of a single distributed system at a consistent level of abstraction chosen by the developer of the description. A precise language, designed for the expression of concepts and functions seen in each viewpoint, is defined in part 2 of the international standards. A graphical notation for the description of components, interfaces, and their interconnection enhances discussion of allocation of security functionality within a system. A

mathematical basis for the reference model is given in part 4 of the standards. Correspondence points in two or more viewpoints help the viewer to determine whether the system is consistent; e.g., security policy should be enforced in a manner consistent with the policy, or similar levels of abstraction are required to achieve precise correspondence. Behavior at some of these points is observable and may be used to assess conformance to the RM-ODP.

Throughout the paper, we illustrate the concepts using a military operations order (OPORD) for a coalition, based on an operations plan (OPLAN). The body of the OPORD informs forces what they must do and when they must do it in coordination with other forces. Various military subcommunities have their own objectives relating to logistics, communications, etc. Each subcommunity has a contribution to the OPLAN, and most sections of the OPLAN have corresponding sections in the OPORD. These sections are produced by different organizations using information of different sensitivities. For example:

- Parts of the personnel and logistics information related to a particular country and service must be restricted to the command staff and to that country and service.
- Communication keys must be restricted to those elements that will be using them.
- Intelligence information must be restricted either to the country producing the information, to a small core of countries, or to the entire coalition.

In this paper, we first define what we mean by "architecture." We then introduce the RM-ODP in greater detail. Then the DGSA is described in terms of the RM-ODP. Finally, we use the RM-ODP to describe federations of U.S. and allied systems in "coalition force systems." A glossary is provided at the end, with more detailed explanations of the terms used in this paper. We also provide a list of abbreviations and acronyms. On finishing the paper, the reader should:

- understand the principles of the DGSA,
- have a rudimentary knowledge of the RM-ODP and understand why it is useful in describing security in distributed systems, and
- begin to understand how an automated system could be designed that would support coalition operations.

## 2. Architecture

The RM-ODP defines a system architecture as

> A set of rules to define the structure of a system and the interrelationships between its parts. [PART2]

While this definition is appropriate in the large, it does not adequately reflect what is necessary for enterprise systems. For this paper, we will use the following definition of architecture, where it is to be understood that we are referring to a *constraint-based architecture*:

> An abstraction for specifying or describing the way a system is organized or constructed using a prioritized list of constraints to determine an acceptable set of implementations. [Leary1995]

Both the RM-ODP and the DGSA may be approached from the standpoint of constraint-based architecture; in both cases, requirements for the systems constrain the resultant implementations.

The importance of architectures is that

> Architectures provide a means to think about something before, during, and after building it. They allow the expression of concepts, structural forms, functions, and properties through the use of views reflecting desirability, possibilities, and constraints. [Mayfield1999]

In particular, architectures facilitate the following:

- High-level reasoning and analysis
- Interoperability of systems
- Portability of software products
- Modularity of software/hardware components
- Replaceability of modules with upgraded versions

An example of an architecture as "constraint tree" may be helpful. Figure 1 on page 6 shows the constraint tree that categorizes an implementation architecture. A tree of possible implementations is generated in which the leaf nodes are the implementations. The other nodes represent decision points where constraints may be imposed.

For example, let $C$ represent application of a constraint and $X$ represent cutting off possible implementations because the constraint governing the marked arc was not selected (at this priority level). In this case, accepting constraint $C2$ followed by constraint $C21$ determined the implementation architecture $I3$, which is the only surviving implementation.

In the constraint tree representation, decisions about which constraint to accept are always ordered by priority as we move from left to right in the tree. If several constraints are of the same priority, the one allowing the largest number of implementations (assuming it is chosen first) is selected as the highest priority constraint within the grouping. If more than one constraint produces the same (largest) number of implementations, any of that group may be selected first.

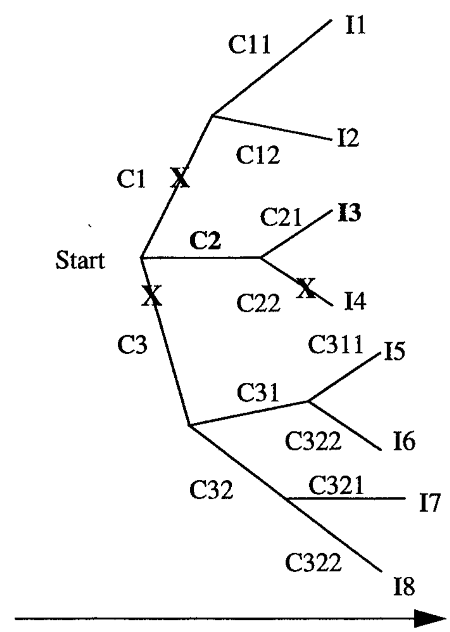



**Figure 1. Definition of an Architecture Family**

The use of such ordered sets of constraints and subsets of constraints provides a method of abstracting from an implementation to an abstract architecture.

### 2.1 Abstraction and Architecture

Current implementations of systems of systems (SoS) contain so much detail that their architectures are rarely captured completely. Because of our inability to perceive the whole, we are then in the situation of those legendary blind men standing around an elephant: one with a hand on the trunk; another with a hand on the tail; a third touching an ear; a fourth touching a leg; and a fifth touching the back. Each has a different grasp on the elephant with a different concept of the whole elephant. Not one of these concepts approximates the whole elephant.

Instead of insisting on using the detailed implementation architecture, one or more abstractions of that implementation are used that focus on problems of interest. For example, if we know that our current bottleneck is in the network between two end systems, we will abstract away the detail that is not relevant to our problem so that the remaining architectural detail is of a size that permits us to focus on the issues deemed responsible for the bottleneck. Care must be used in the abstraction process so that "important features" are not abstracted away.

In terms of the constraint-based definition of an architecture, this amounts to a selection (as a partial order) of the constraints in their original priority order:

- Only those constraints that are relevant to the problem of interest are included, and
- Only the detail required by those constraints needs to be examined.

Such architectures are not implementation architectures but are *abstractions* of implementation architectures. As in the example of the elephant, there may be many abstractions related to a given implementation. In Figure 1, if we do not choose to accept constraint *C21*, thus cutting off constraint *C22*, we find that two implementations could satisfy the set of constraints which we are using. Their common features categorize this abstraction of the architecture.

The DGSA notes four levels of architectural specification. (The numbers refer to sections of the DGSA document.)

> **Level 1: Abstract Architecture.** "Starts with requirements and defines functions to be performed." "Cites principles, fundamental concepts, and functions that satisfy the typical security requirements." "Concepts and functions are allocated to elements of an abstract definition of information systems architecture." [DGSA 1.3.1]
>
> **Level 2: Generic architecture.** "Proceeds from an initial allocation of security services and functions and begins to define the types of components and security mechanisms that are available to implement the security services with particular strength." [DGSA 1.3.2]
>
> **Level 3: Logical architecture.** "Subjects a generic architecture to hypothetical requirements resulting in a detailed example (no cost analysis required)." [DGSA1.3.3]
>
> **Level 4: Specific architecture.** "Uses real requirements to permit acquisition or development by component." "[A]ddresses components, interfaces, standards, performance, and cost." [DGSA 1.3.4]

Examining these architectural specification levels from level 1 to level 4, we see that from the standpoint of constraints, constraints are added to 1 to produce 2; to 2 to produce 3; and to 3 to produce 4. We assume that a specific architecture has one implementation. We then observe that the number of implementations, I(level) is

$$I(4) = 1$$

and

$$I(1) \geq I(2) \geq I(3) \geq I(4)$$

Unfortunately, this notion of an architecture based on constraints does not have sufficient descriptive power to deal with systems of systems, largely because of the wide variety of con-

cerns with which we might wish to deal and the overloading of words with which different communities describe these concerns. A mutually agreed language and semantics for expressing a set of related concerns and a commonly assumed taxonomic model (abstraction) of those concerns are required. In the standards community, these abstractions are called *reference models.*

### 2.2 DoD Requirements[1]

In 1993, prior to the development of the DGSA, a new set of requirements was identified that lead to a major alteration in DoD's approach to information security [NSA93]. DoD information systems must satisfy requirements 1 through 4:

> **Requirement 1.** Support information processing under multiple security policies of any complexity or type, including those for sensitive unclassified information and multiple categories of classified information.
>
> **Requirement 2.** Be sufficiently protected to allow distributed information processing (including distributed information system management) among multiple hosts on multiple networks in accordance with open system architectures.
>
> **Requirement 3.** Support information processing among users with different security attributes employing resources with varying degrees of security protection, including users of non-secure resources if a particular mission so dictates.
>
> **Requirement 4.** Be sufficiently protected to allow connectivity via common carrier (public) communications systems.

Further supporting these requirements, the Chairman of the Joint Chiefs of Staff recently issued a policy with respect to Defensive Information Operations [CJCSI1997]. This policy requires that Defense Information Systems shall address the following:

> **Policy a.** Information, information-based processes, and information systems (such as command, control, communications, and computer (C4) systems, weapon systems, and infrastructure systems) used by US military forces will be protected relative to the value of the information contained therein and the risks associated with the compromise of or loss of access to the information.
>
> **Policy b.** Information system defense relies on four interrelated processes. These include a process to protect information and information systems, a process to detect attacks or intrusions, a restoration process to mitigate the effects of incidents and restore services, and a response process.

1. Material in these first sections is drawn from [FM98]. After DGSA was promulgated, DoD agencies adopted a Defense in Depth strategy to help meet QoP concerns.

These requirements reflect a clearer understanding of today's information protection needs in both defense and commercial communities.

### 2.2.1 Consequences of Requirements

Requirements 1 through 4 and Policies a and b result in three important changes in the way in which DoD protects its information systems. These changes are discussed in the following subsections.

#### 2.2.1.1 Use of Public Networks

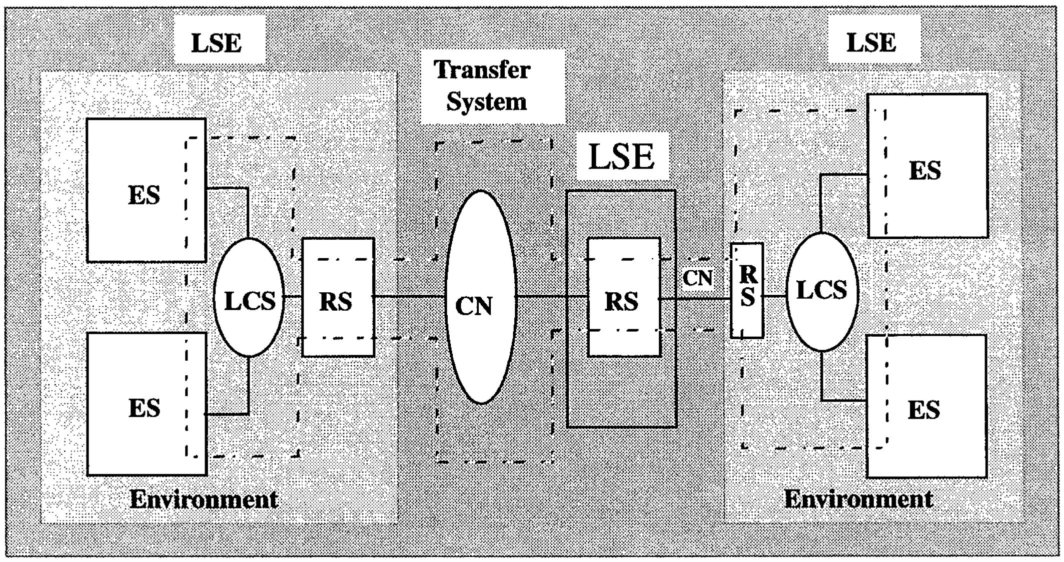
The first important change involves DoD's use of networking (see Requirement 4). DoD is moving away from the use of dedicated circuits to the use of the public-switched network that it cannot control. This means that expectations of security must be placed at the end points of the network, and that only availability can be demanded from public network providers. We shall use the term *communication network* (CN) hereafter to refer to both dedicated and public wide-area networks.

The DoD end systems are the responsibility of the Department—they are physically and logically under DoD control. The first step then in protecting such systems is to provide a safe, protected environment in which they can reside (i.e., an *enclave*). We shall call such an enclave a *local subscriber environment* (LSE). Finally, when we discuss information transfer, we include some portion of the enclaves at both ends of a network, relay systems within a network, and the network itself (see Figure 2 on page 10). We shall refer to this combination as the *transfer system*, which we also need to protect in distributed computing.

#### 2.2.1.2 Multiple Security Policies

The second important change is that DoD historically has primarily been concerned with confidentiality policies [ClarkWilson], but now it must also accommodate integrity, availability, and accountability security policies (see Requirement 1 on page 8). DoD has much in common with commercial business enterprises. It has personnel, pay, inventory, purchasing, accounting, and invoicing functions that have their own unique protection policies. The Department also has many unique policy elements, e.g., protection of classified information, although this is not totally unlike protecting highly sensitive corporate information. This type of classification-based confidentiality policy, loosely stated, is that to read data, a user must be cleared for the level of sensitivity of the information and also must have a documented "need to know" the information.

Many DoD systems must handle a variety of types of sensitive information and, therefore, must be capable of enforcing multiple security policies. For example, a bomber needs to




Source: [DGSA]

——— communications path

- - - - - transfer system boundary

CN communications network
ES end system
LCS local communication system
LSE local subscriber environment
RS relay system

**Figure 2. Conceptual View of an Information System**

maintain both weather information, for which integrity is critical, and targeting information, for which confidentiality is critical. The policies define a partitioning of the information in these systems: information must be identified with the applicable policy and enforcement mechanisms appropriate to the policy used for accesses to that data.

The aggregate security policy applicable to information in a DoD system frequently is not static but changes over time. Such policy must adapt to new laws that are passed or executive orders that are issued. Also, this policy must adapt to changing conditions such as the formation of a coalition, the start of a conflict, or the detection of an attack. With a change in policy, the application of security mechanisms to information may change. Therefore, security enforcement mechanisms should be policy neutral—they should not embody any particular policy [DTOS].

#### 2.2.1.3 Sufficient Protection

A third important change is found in the use of the words "sufficiently protected" in Requirements 2 and 4 (page 8). Previously, DoD concentrated on the task of protecting systems against all attacks. The protection mechanisms were assured to work at some level of confidence commensurate with the sensitivities of the data that could be processed by the sys-

tem. The agency responsible for the data stored on the system then accredited the system for use in a particular operational environment through a certification process.

In this new set of requirements, DoD focuses on risk management. The strength of defensive mechanisms required depends on the value of the information to the mission and the perceived threat (see Policy a on page 8). Additionally, DoD recognizes that protection is not absolute and the protection mechanisms might be breached (see Policy b on page 8). DoD's policy now is "protect, detect, react, restore, and respond." The effort put into each phase is related to the value of the information to be protected and the cost if the information is compromised or destroyed. Recognizing that no security is perfect, a risk assessment is made as to whether the protection provided is "sufficient."

### 2.2.2 Additional Requirements

There are additional requirements that result from economic realities and the needs of the missions that use information. These are discussed separately in the following subsections.

#### 2.2.2.1 Use of COTS and GOTS Software

DoD recognizes that it needs to purchase and use commercial-off-the-shelf (COTS) and government-off-the-shelf (GOTS) software in order to provide functionality that its elements need and expect. Further, DoD has accepted the fact that it does not constitute a large enough market to independently cause commercial developers to provide advanced technology implementations in a timely fashion with all the desired assured security features. Nor does DoD have the resources to concurrently re-engineer its legacy applications to provide them with such security features. The ability to use this off-the-shelf software without change, but in an encapsulated environment where the security infrastructure assures that the security policy is not violated, is required. Under the assumption that an arbitrary security policy is used, meeting this requirement is especially difficult.

#### 2.2.2.2 Information Centralization, Access, and Interoperability

In the documents describing Joint Vision 2010, the Services visualize an information system encompassing all aspects of warfare that is centrally administered and maintained, but is universally accessible and usable on any type of machine in the network [JV2010].

#### 2.2.2.3 Total Access to All Necessary Information

The Joint Vision 2010 documents state that an appropriately cleared person must have access to all information needed to plan and execute his mission [JV2010]. This implies that the systems containing the information are available.

#### 2.2.2.4 Information Separation

Regardless where information is stored, it is to be kept separate from all other information in different protection or policy regimes.

#### 2.2.2.5 Increased Connectivity Without Increased Cost

This is to be achieved by use of commercial infrastructures that can provide the additional connectivity at no increase in cost.

#### 2.2.2.6 Increased Access to Information and Resources

Military efficiency in the past has been lessened because those who need access could not obtain it. The new systems must provide interoperability of communications and security services to permit this improved access and management of security features.

#### 2.2.2.7 Transparency in Distributed Processing

Because information may be placed anywhere or moved anywhere in a network, it is important that those who wish to access it need not know the physical location of the information or of the physical address at which it resides.

#### 2.2.2.8 Consistent and Uniform Certification and Accreditation

DoD has *many* systems. They are continually being redeployed and attached to networks to which they have not been previously attached. Unless waivered, they may not be "turned on" until the appropriate agencies have certified and accredited them in this new environment.

### 2.2.3 Variety of Security Policies Supporting the Requirements

Security requirements become increasingly concrete as the level of design becomes more detailed, moving from high-level policy to services to particular mechanisms. As the policy is represented using services, decisions are made and the architecture becomes more constrained. Likewise, when services are represented using specific mechanisms, the architecture again is constrained.

In an abstract architecture, DoD is concerned with confidentiality, integrity, availability, and accountability security policies. The strength of security is also stated, based on the value of the information to the mission and the strength of the threat. A generic, parameterized policy might be as follows:

> A {document} may only be modified by an {authorized individual1} and must be signed by that modifying individual. It may only be viewed by {authorized individual2} and United States personnel designated by {authorized individual2} for {period1} after being produced; thereafter it may be read by

> anyone. Modifications must be available within {period2} of being made and signed. The strength of security mechanisms must be appropriate for a threat with {resource strength} and {damage1} damage to the mission by undetected unauthorized modification of the {document}, {damage2} damage for unauthorized viewing, and {damage3} damage for loss of availability.

For an OPORD, the actual policy might be as follows:

> The intelligence annex may only be modified by an intelligence officer, and must be signed by that modifying individual. It may only be viewed by commanders and designated United States personnel designated by those commanders within the 48-hour period after being produced; thereafter it may be read by anyone. Modifications must be available within 15 minutes after being made and signed. The strength of security mechanisms must be appropriate for a threat with access to the system and grave damage to the mission by undetected unauthorized modification of the intelligence annex, severe damage for unauthorized viewing, and severe damage for loss of availability.

Thus, the confidentiality requirement is that the annex be able to withstand unauthorized insider attempts to read it for 48 hours. The integrity requirement is that only the intelligence officer can modify the annex. The accountability requirement is that one can determine all of the individuals who contributed to the annex.

At a more detailed level of design, the policy and threats are translated into requirements for security services that are to be provided by the software and hardware of the system. Doing so produces a generic architecture that constrains the set of acceptable designs to those that provide the specified services. One categorization of services is the Common Criteria functional classes: audit, communication, cryptography, access control, identification and authentication, security management, privacy, protection of security services, resource utilization, system access, and trusted path [CC]. For example, to assure the confidentiality of the OPORD, the following services might be required:

- Cryptography with strength sufficient so that unauthorized users of the organization's system will not be able to break it within 48 hours.
- Authentication sufficiently strong to prevent masquerading.
- Access control capable of protecting private keys.
- Trusted path for authentication and key dispersal.

Authentication and cryptography will also be useful for the accountability requirements. In general, a high-level security requirement will be supported by multiple security services and a security service will support multiple requirements.

The next level of design provides security services using specific mechanisms, producing a specific architecture with very few choices. The cryptography service might be satisfied by triple Data Encryption Standard (DES), while authentication might be satisfied using smart cards. An information system need not be homogeneous: some nodes might instead use retinal scans for authentication.

# 3. RM-ODP

In this chapter we briefly describe portions of the RM-ODP that are relevant to our description of DGSA. Those interested in a complete description are encouraged to read the specification [PART1, PART2, PART3, PART4] and the detailed examples found in Blair and Stefani's *Open Distributed Processing and Multimedia* [BS1998]. This book describes how RM-ODP may be used to describe multimedia designs and develops extensions to the Object Management Group's Object Request Broker (ORB) that are necessary to implement the system.

Other uses of RM-ODP are found in the Telecommunications Intelligent Networking Architecture (TINA - C) and in the International Telecommunications Union (ITU) specifications for management in the ITU-T X.7xx (and their corresponding ISO/IEC specifications). (In the next chapter we will give examples of how RM-ODP may be used to describe the DGSA.)

As mentioned in the introduction, the RM-ODP addresses concerns of enterprise systems through five viewpoints and five specialized languages:

- Enterprise viewpoint: purpose for, scope of, and policies for activities performed by the system.
- Information viewpoint: information object and information processing semantics.
- Computational viewpoint: objects, interfaces, and configurations.
- Engineering viewpoint: mechanism and functions required to support distributed interactions between objects.
- Technology viewpoint: ODP standards and products to be used.

The RM-ODP was developed over a very long period by international standards committees determined to deal with enterprise concerns, including several distribution transparencies:

- Access transparency: "masks data representation & invocation mechanisms."
- Failure transparency: "masks from an object, the failure and possible recovery of other objects or itself to enable fault tolerance."

- Location transparency: "masks the use of information about location in space when identifying and binding to interfaces."
- Migration transparency: "masks, from an object, the ability of a system to change the location of that object."
- Persistence transparency: "masks, from an object, the deactivation and reactivation of other objects or itself."
- Relocation transparency: "masks relocation of an interface from other interfaces bound to it."
- Replication transparency: "masks the use of a group of mutually behaviorally compatible objects to support an interface."
- Transaction transparency: "masks coordination of activities among a configuration of objects, to achieve consistency."

Aspects of a system of systems are abstracted away if they do not affect the achievement of these transparencies under conditions engendered by requirements in one of the five viewpoints.

The committees experimented with sets of viewpoints to see if they could determine a "minimal set" that covered all the transparencies they wished to provide in an enterprise-wide system of systems. They concluded that although the five viewpoints described above might not be the only set of viewpoints that could be used, this set was sufficient. It appeared that areas of concern could be composed in terms of the five enterprise languages in enough detail to describe solutions or to answer questions of interest. For example, we might wish for a security viewpoint in addition to those chosen that might more directly express the concepts of the DGSA. As will be seen later in this chapter, the concepts of the DGSA may be expressed in terms of the other viewpoints.

The committees then concentrated on languages to express concepts in each of the viewpoints. While some of the terms they selected describe concepts in multiple viewpoints, other terms are viewpoint specific. The terms were then used to develop transparency prescriptions for distribution transparencies indicated above, e.g., location transparency as required for the viewpoint. Specification of functions required to support ODP systems in each viewpoint was developed based on the needed transparencies, e.g., bridging of the use of security technologies.

Because a system of systems (SoS) is described by applying descriptions from the five viewpoints simultaneously, there are correspondence points between descriptions in two or

more viewpoint. For example, a security management policy for an SoS as expressed in an enterprise language from an enterprise viewpoint would be reflected in its implementation as seen in an engineering viewpoint and as expressed in the engineering language. If actual products were used that corresponded to components used in a system, e.g., Tivoli network management, these products would have correspondence points in the engineering viewpoint and in the technology viewpoint.

Correspondence points occur at interfaces in the system. At some correspondence points behavior can be observed. This behavior can be described in the languages of the viewpoints in which the correspondence points lie. Conformance to the RM-ODP specifications can be tested based on these observed behaviors. Some behaviors and some system states are not observable. Conformance statements cannot be based on these behaviors or system state.

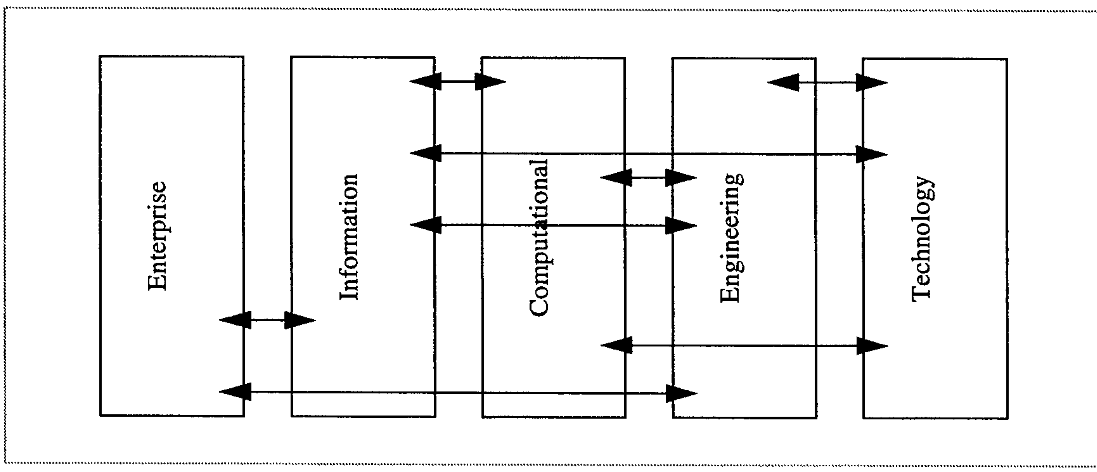



**Figure 3. Examples of Conformance Points**

The viewpoint languages vary in the richness of concept expressible. Some have been developed more fully than others. The richest expression is probably in the engineering language. Recently, ISO working groups have endeavored to develop the enterprise language to deal with business objects [ISO15414], so this relative "lead" of the engineering language may not last long. Tables 1 through 5 indicate the concepts of the languages of the various viewpoints using the vocabulary of [PART2].[1]

---

1 Most of the material in Tables 1 through 5 is directly quoted from [PART3], with some material paraphrased for the sake of brevity. See [PART2] and [PART3] for further details.

**Table 2. Concepts of the Enterprise Language**

| Term | Concept |
|---|---|
| Activity | A single-headed acyclic graph of actions, where occurrences of each action in the graph is made possible by the occurrence of all immediately preceding actions. |
| Action | Something that happens. Each action is defined by the action name and the specification of the action policy. |
| Behavior (of an object) | A collection of actions with a set of constraints on when they may occur. Constraints may include, for example, sequentiality, non-determinism, concurrency, or real-time constraints. |
| Community | A configuration of objects formed to meet an objective. The objective is expressed as a contract that specifies how the objective can be met. |
| Contract | An agreement governing part of the collective behavior of a set of objects. A contract places obligations on the objects involved. The specification of a contract may include a specification of the different roles that objects involved in the contract may assume, and the interfaces associated with the roles; quality of service attributes; indications of duration or periods of validity; indications of behavior that invalidates the contract; and liveness and safety conditions. |
| <X> Domain | A set of objects, each of which is related by a characterizing relationship <X> to a controlling object, e.g., naming domain, management domain, information domain.[a] |
| Environment (of an object) | The part of the model that is not part of that object. |
| <X> Federation | A community of <X> domains. |
| Obligation | A prescription that a particular behavior is required. An obligation is fulfilled by the occurrence of the prescribed behavior. |
| Objective (of a system) | The purpose of the system, stated as the community contract of the <s> community (that specifies how the objective can be met). |
| Permission | A prescription that a particular behavior is allowed to occur. A permission is equivalent to there being no obligation for the behavior not to occur. |
| Policy | A set of rules related to a particular purpose. A rule can be expressed as an obligation, a permission, or a prohibition. |
| Principal Role | A role that includes agreement to be bound by a contract for the community concerned. |
| Process | A collection of actions taking place in a prescribed manner and leading to the accomplishment of some result. |
| Prohibition | A prescription that a particular behavior is prohibited. A permission is equivalent to there being an obligation for the behavior not to occur. |
| Resource | An object modeling an entity that may become unavailable because it is allocated or used up. |
| Role | Identifier for a behavior, which may appear as a parameter in a template for a composite object and be associated with one of the component objects of the composite object. |
| Scope (of an ODP system) | The set of all statements about the roles of the system when seen as a single object in each of the communities of which it is a member. |

a. **Note:** <X> is a syntactic placeholder for kind of domain, e.g., administrative.

**Table 3. Concepts of the Information Language**

| Term | Concepts |
|---|---|
| Invariant Schema | A set of predicates on one or more information objects that must always be true. The predicates constrain the possible states and state changes of the objects to which they apply. |
| Static Schema | A specification of the state of one or more information objects, at some point in time, subject to the constraints of any invariant schema. |
| Dynamic Schema | A specification of the allowable state changes of one or more information objects, subject to the constraints of any invariant schema. |

**Table 4. Concepts of the Computational Language**

| Term | Concepts |
|---|---|
| Signal | An atomic shared action resulting in one-way communication from an initiating object to a responding object. |
| Operation | An interaction between a client object and a server object that is either an interrogation or an announcement. |
| Announcement | An interaction—the *invocation*—initiated by a client object resulting in the conveyance of information from that client object to a server object, requesting a function to be performed by that server object. |
| Interrogation | An interaction consisting of an invocation followed by a second interaction—the *termination*—initiated by the server object, resulting in the conveyance of information from the server object to the client object in response to the invocation. |
| Flow | An abstraction of a sequence of interactions, resulting in conveyance of information from a producing object to a consumer object. |
| Signal Interface | An interface in which all interactions are signals. |
| Operation Interface | An interface in which all interactions are operations. |
| Stream Interface | An interface in which all the interactions are flows. |
| Computational Object Template | An object template that comprises a set of computational interface templates that the object can instantiate, a behavior specification, and an environment contract specification. |
| Computational Interface Template | An interface template for either a signal, stream, or operation interface. A computational interface template comprises a signal, a stream, or an operation interface signature as appropriate; a behavior specification; and an environment contract specification. |
| Binding Object | A computational object that supports a binding between a set of other computational objects. |

**Table 5. Concepts of the Engineering Language**

| Term | Concepts |
|---|---|
| Basic Engineering Object | An engineering object that requires the support of a distributed infrastructure. |
| Cluster | A configuration of basic engineering objects forming a single unit for the purposes of deactivating, checkpointing, reactivation, recovery, and migration. |
| Cluster Manager | An engineering object that manages the engineering objects in a cluster. |
| Capsule | A configuration of engineering objects forming a single unit for the purpose of encapsulation, processing, and storage. |
| Capsule Manager | An engineering object that manages the engineering objects in a capsule. |
| Nucleus | An engineering object that coordinates processing, storage, and communications functions for use by other engineering objects within the node to which it belongs. |
| Node | A configuration of engineering objects forming a single unit for the purpose of location in space that embodies a set of processing, storage, and communication functions. |
| Channel | A configuration of stubs, binders, protocol objects, and interceptors providing a binding between a set of interfaces to basic engineering objects through which interaction can occur. |
| Stub | An engineering object in a channel that interprets the interactions conveyed by the channel, and performs any necessary transformation or monitoring based on this interpretation. |
| Binder | An engineering object in a channel that maintains a distributed binding between interacting basic engineering objects. |
| <x> Interceptor | An engineering object in a channel, placed at a boundary between <x> domains that bridges the two domains, e.g., security, technology. |
| Protocol Object | An engineering object in a channel that communicates with other protocol objects in the same channel to achieve interaction between basic engineering objects. |
| Communications Domain | A set of protocol objects capable of interworking. |
| Communication Interface | An interface of a protocol object that can be bound to an interface of either an interceptor object or another protocol object at an interworking reference point. |
| Binding End Point Identifier | An identifier in the naming context of a capsule, used by a basic engineering object to select one of the bindings in which it is involved, for the purpose of interaction. |
| Engineering Interface Reference | An identifier, in the context of an engineering interface reference management domain, for an engineering object interface that is available for distributed binding. |

**Table 5. Concepts of the Engineering Language (Continued)**

| Term | Concepts |
|---|---|
| Engineering Interface Reference Management Domain | A set of nodes forming a naming domain for the purpose of assigning engineering interface references. |
| Engineering Interface Reference Management Policy | A set of permissions and prohibitions that govern the federation of engineering interface reference management domains. |
| Cluster Template | An object template for a configuration of objects and any activity required to instantiate those objects and establish initial bindings. |
| Checkpoint | An object template derived from the state and structure of an engineering object that can be used to instantiate another engineering object, consistent with the state of the original object at the time of checkpointing. |
| Checkpointing | Creating a checkpoint. Checkpoints can only be created when the engineering object involved satisfies a pre-condition stated in a checkpoint policy. |
| Cluster Checkpoint | A cluster template containing checkpoints of the basic engineering objects in a cluster. |
| Deactivation | Checkpointing a cluster, followed by deleting of the cluster. |
| Cloning | Instantiating a cluster from a cluster checkpoint. |
| Recovery | Cloning a cluster after cluster failure or deletion. |
| Reactivation | Cloning a cluster following its deactivation. |
| Migration | Moving a cluster to a different capsule. |

**Table 6. Concepts of the Technology Language**

| Term | Concern |
|---|---|
| Implementable Standard | A template for a technology object. |
| Implementation | A process of instantiation whose validity can be subject to a test. |
| IXIT | Implementation eXtra Information for Testing. |

Each language has a series of concepts, a set of structuring rules, and a set of conformance and reference points. For example, the structuring rules of the engineering language consist of the following:

- Channel rules
- Interface reference rules
- Distributed binding rules
- Relocation rules
- Cluster rules
- Capsule rules
- Node rules
- Application management rules
- Failure rules

We will exhibit some of these rules in an example later in Chapter 5.

The observant reader will notice that each concept is required because of one of the transparency considerations and/or because of the need to manage the SoS. Further, the scheme is designed for a dynamic environment. Engineering objects, collections of objects, capsules containing objects, and channels between capsules are dynamically instantiated, employed, and destroyed. It is clear that the RM-ODP provides many needed concepts required in order to describe SoS needed by the DoD enterprise. Let us examine the principles behind the RM-ODP in greater detail.

### 3.1 The Enterprise Viewpoint

An enterprise specification represents an information system and its environment as a *community*. Within a community, each object fulfills at least one role that identifies a behavior of the object. Objects are acted upon within the community by actions; a behavior is a set of actions with constraints on when each can occur. An object that participates in an action is called an *actor*, and an object that is referenced in an action is called an *artifact*. The occurrence of an action is governed by a policy, or set of rules, in addition to the constraints. A rule is either:

- an *obligation*,
- a *permission*,
- a *prohibition*, or
- a *negation* of an obligation, permission, or prohibition.

A community, then, is specified as follows:

- an objective,
- a set of roles played in reaching the objective, and
- a policy governing how the objective may be reached.

Communities may be composed of sub-communities and may be created or destroyed as required. A community is established by a contract that is an agreement to an objective by a set of objects.

An *action template* specifies a policy for the action and role required to perform the action. At least one principal with the role of actor is associated with each action. When the template is instantiated, the actor(s) performing the action are selected and given the required role nec-

essary to perform the action. The artifacts and resources required to complete the action are also identified and allocated. After the action has taken place, unused resources are released.

An *activity template* specifies an activity policy, roles required to complete the activity, and a partially ordered set of actions that must be accomplished in order to complete the activity. When instantiated, the actor(s) performing the activity are selected and assume the required role necessary to perform the activity. The artifacts and resources required to complete the activity are also identified and allocated. After the activity has taken place, unused resources are released.

A *process* is a set of interrelated activities, possibly performed in different communities. For example, consider a group that must produce a report on a subject. They constitute a community whose objective is to produce the report. The community may have a variety of roles such as researcher, writer, editor, and publisher. There may be many aspects of the policy of the community that are employed in reaching the objective, e.g., no part of the report is to be released to other communities except by the principal with the role of publisher. The activities are usually related to the roles, e.g., editing the document. Members of the group are the principals who agree to the portions of the contract, e.g., the report is due on April 27th, 2000. Resources are assigned so that the community can complete its objective, e.g., only the writer may access certain files until a draft is complete. When the community is finished producing the report, it can be redeployed and/or reallocated.

### 3.2 The Information Viewpoint

The information viewpoint contains information objects and invariant, static, and dynamic schemata that relate them:

- An *invariant schema* is a set of predicates on information objects that must always be true. It constrains the possible states and state changes of the objects to which it applies. The state describes objects at some point in time in terms of their values, their attributes and attribute values, and their relationships with other objects; the current state is determined by the initial state and the sequence of actions that have occurred.
- A *static schema* specifies the state of one or more information objects.
- A *dynamic schema* represents the change of state and/or functional change produced by an action. This change can result in creation of new instances of objects, modification of instances of objects, or deletion of instances of objects; creation of new attributes for an object or deletion of old attributes for an object; creation of

new relationships and or associations[2], and deletion of old relationships or associations. The dynamic schema represents the ability of the SoS to change over time.

Each community in the enterprise viewpoint is associated with a set of invariant, static, and dynamic schemata. Each instance of an action in the enterprise viewpoint is linked to a static schema that represents the state at the beginning of the action. It "causes" the change as exhibited in an instance of a dynamic schema, which results in a new static schema representing the state at the end of the action.

As an example, consider a bank account. One information object might be *balance*. An invariant schema might specify that the value of *balance* is a non-negative amount of money. At some point in time, a static schema asserting *balance*=$500 might be true. A dynamic schema associated with a withdrawal of $100 would be *balance'*=*balance*-$100 (where *balance* is the value before the action, and *balance'* is the value afterwards). This would cause the static schema *balance*=$400 to become true. This dynamic schema could not be used at a time when *balance*=$50 because the result would violate the invariant schema.

### 3.3 The Computational Viewpoint

A computational object encapsulates data and its processing, and offers interfaces to other objects. Figure 4 represents an object that interacts with some environment. All interactions of the object—its flows, operations, and signals—are via its interfaces; the interfaces partition the interactions. From the object's point of view, there are only two entities: itself and the environment as perceived through its interfaces. (This is also true for basic engineering objects.) An object may have more than one interface and may dynamically create new interfaces or destroy old interfaces. These interfaces are bound to or unbound from interfaces of other objects.[3]

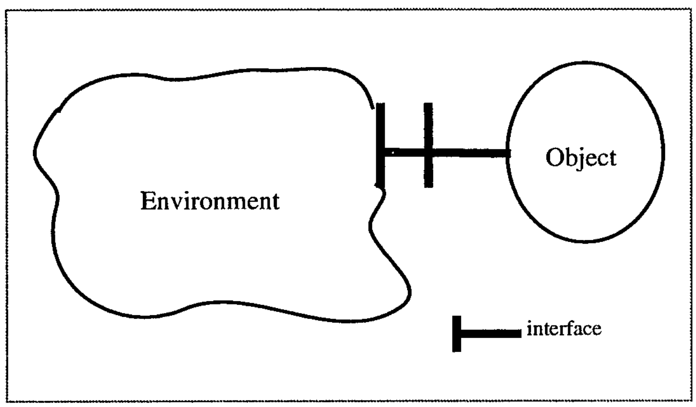


**Figure 4. Basic RM-ODP**

2 See ISO10165-7 for examples.

3 A new or deleted interface may imply that a new schema is in place.

The computational viewpoint provides the transparencies described in the introductory material of Chapter 3. Thus the binding object in Figure 5 provides the required connections between the interfaces of objects B and A whether they are on the same system or different systems and whether those systems are heterogeneous or homogeneous. Note that the binding object provides a multiparty binding. The management object represents the fact that the binding object may be instantiated, bound to objects A and B, unbound, and deleted under control of elements of the system. It is required to represent the fact that the connection of object interfaces *may* be highly dynamic.

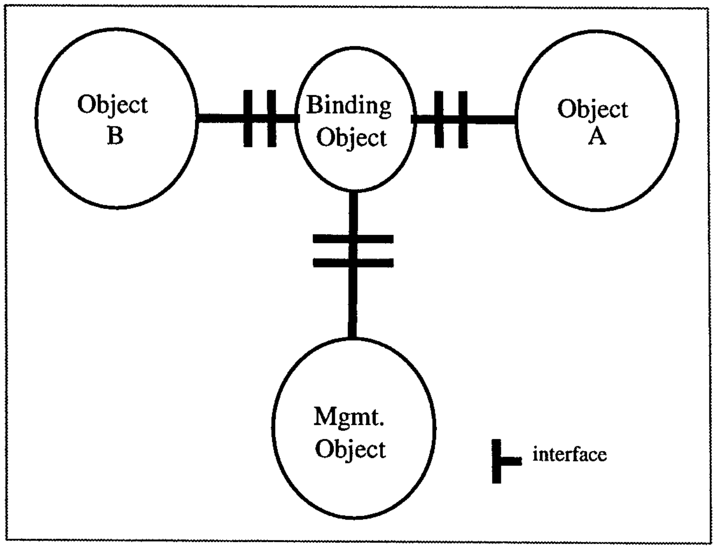



**Figure 5. Basic Objects in the Computational Viewpoint**

### 3.4 The Engineering Viewpoint

The engineering viewpoint must show the infrastructure. The simplest realization of the computational viewpoint shown previously is to have objects A and B within a cluster within a capsule supported by a nucleus on a single node (Figure 6 on page 26). The engineering viewpoint may require that many engineering objects be used to represent this situation because basic objects representing infrastructure must be shown.

In the engineering viewpoint, the computational objects A and B are represented (in this instance) by basic engineering objects $E_A$ and $E_B$ that are on a single node and in a single capsule. This means that they use "local binding" instead of the "distributed binding" that they would use if the two objects were in different capsules (possibly on different nodes). The node is controlled by a nucleus that supports a capsule containing the engineering objects and their managing objects: a capsule manager and a cluster manager, as well as a channel that connects to the "outside world."

A scenario for creation might be as follows:

1. The nucleus creates the capsule and capsule manager from their templates, binding itself to the capsule manager.

2. The capsule manager creates the cluster manager and the channel from their templates, binding itself to both.

3. The cluster manager creates the engineering objects from their templates, binding itself to both $E_A$ and $E_B$, causing them to be initialized and started.

4. The engineering objects request that they be bound to:

   - one another,
   - the nucleus, and
   - the channel.

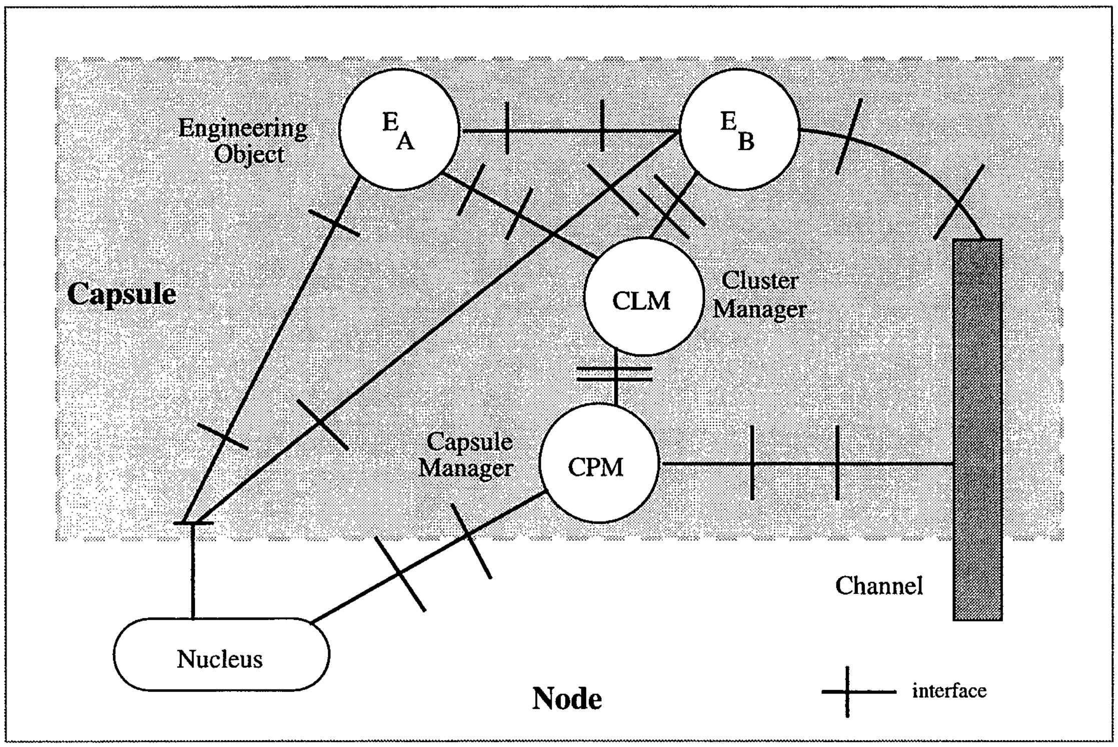



**Figure 6. Engineering Viewpoint of A & B**

In this case the engineering viewpoint's binding object is realized as a local binding.

In practical terms, we make the following analogies:

- The nucleus is like what we think of as an operating system or operating system kernel.
- A capsule is like a process container with an address space that is isolated from all other address spaces on the platform.
- The capsule manager is like a main Unix process.

- The cluster manager is like the mechanism that is used to support Dynamic Link Libraries (DLL) in Microsoft Windows.
- And the channel is like a group of Unix STREAM modules (as we shall see when we examine it in greater detail).

All in all, these few engineering modules provide an appropriate set of components for constructing arbitrary applications and for allocating security services among them.

The computational and engineering viewpoints are obviously closely related by direct correspondences. The other three viewpoints are related to these two but not as directly. For example, the enterprise viewpoint describes the management policies for the capsule, including any security policy that must be interpreted or enforced. The information viewpoint would present the schema for the information in the joint world of objects A and B, showing relationships and associations. Because A and B are closed to one another except by means of the interfaces that they supply, this schema may be in two parts: A's information and operations and B's information and operations. Additionally, there may be "forwarding operations" that permit information from A to be forwarded to B and vice-versa.

Now let us consider how the engineering viewpoint changes if we assume that object A and object B are on different platforms and have used "distributed binding" instead of local binding. The resultant is illustrated in Figure 7 on page 28. One possible scenario for generating this system is as follows:

1. Nuclei of nodes A and B instantiate capsules and capsule managers.
2. The capsule managers instantiate clusters and cluster managers from their templates.
3. The cluster managers instantiate the engineering objects $E_A$ and $E_B$ that are registered with their nuclei through the cluster managers and capsule managers.
4. One of the two objects requests the capsule manager to set up a channel between itself and the other engineering object.
5. One of the nuclei sets up a channel controller that instantiates two stubs, two binders, two protocol objects, and one interceptor. (Note that an interceptor is only required if there is some form of heterogeneity between node A and node B, e.g., security policy, technology, management domain.)
6. The channel controller causes the contents of the channel to be bound together and returns interface references to the capsule managers that cause the engineering objects to be bound to the stubs of the channel.

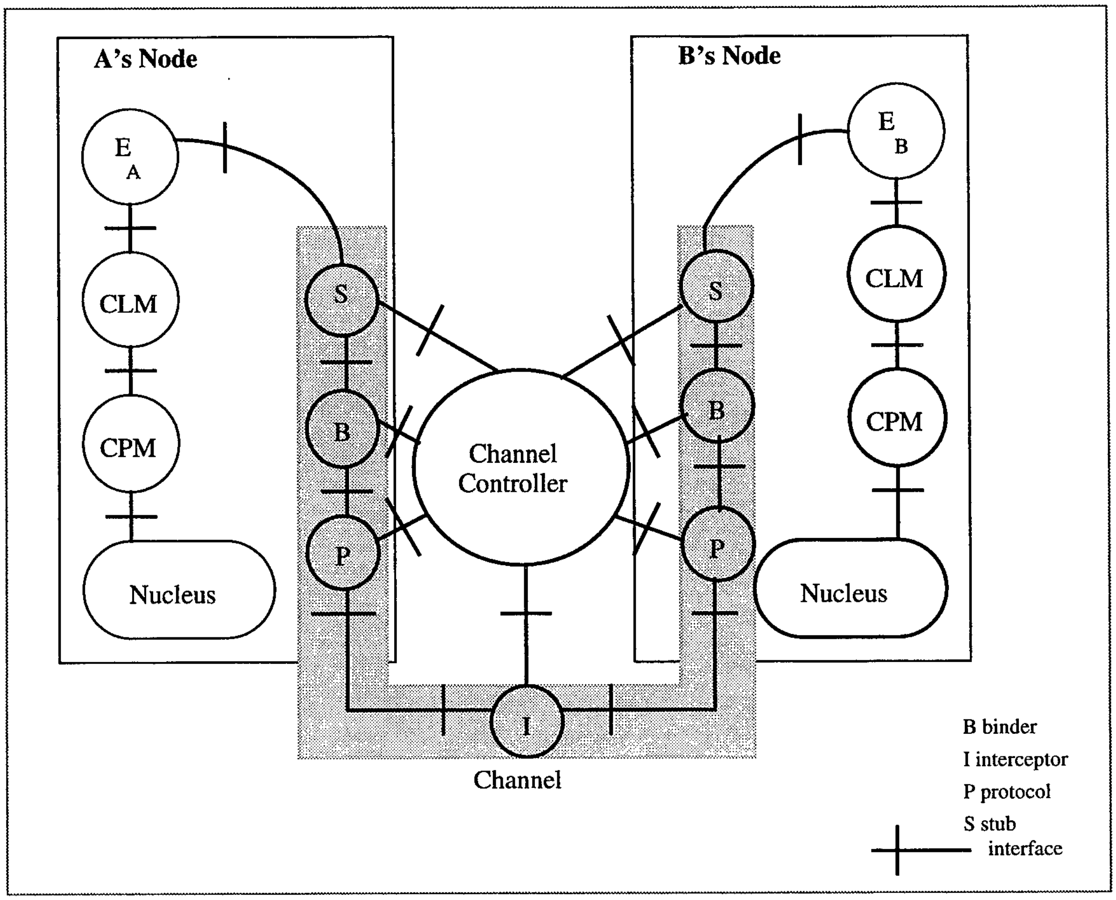



**Figure 7. Distributed Objects Bound Together**

In this case, the engineering object corresponding to the computational viewpoint binding object is realized as a channel.

Notice that the channel and its controller abstract away all issues of the communications reference model that are not directly relevant to the list of transparencies outlined previously. Any communications stack that permits uni- or bi-directional communications is sufficient. The engineering viewpoint does not present the entire picture, however. The enterprise viewpoint may present requirements on the communication between the two applications objects. The information viewpoint may place requirements on the information model relating the two kinds of information.

### 3.5 The Technology Viewpoint

A technology specification defines the choice of technology for an ODP system. The technology language is used to assert that technology objects are instances of implementable standards, which usually contain conformance statements.

The technology viewpoint could deal with such issues as what operating system provides the nucleus functions. The technology viewpoint may require use of a specific set of product standards for interoperation, e.g., TCP/IP (Transmission Control Protocol/Internet Protocol) and RPC[4] (Remote Procedure Call).

### 3.6 Mapping Between Viewpoints

For the SoS to be consistent, corresponding points in the viewpoints must be consistent. Figure 8 on page 30, reproduced from an early version of Annex C of [ISO15414], shows the influence of objects in a viewpoint on objects in other viewpoints.

We note that for each elemental action (presumably carried out in a capsule) there must exist a set of schemata in the information viewpoint which are used or affected by the action. When the primary focus of concern is security, at some level of abstraction the information viewpoint will show the information entities that could be utilized or affected and all functionality available to the actor utilizing the capsule.

The mapping between computational and engineering viewpoints is much more direct. Therefore it is not illustrated.

In the next chapter we will consider what must be elaborated as we add technical security to the list of concerns to be dealt with in our SoS.

---

4 That is, whether the Object Management Group's CORBA or The Open Group's Distributed Computing Environment provide the infrastructure required for location transparency.

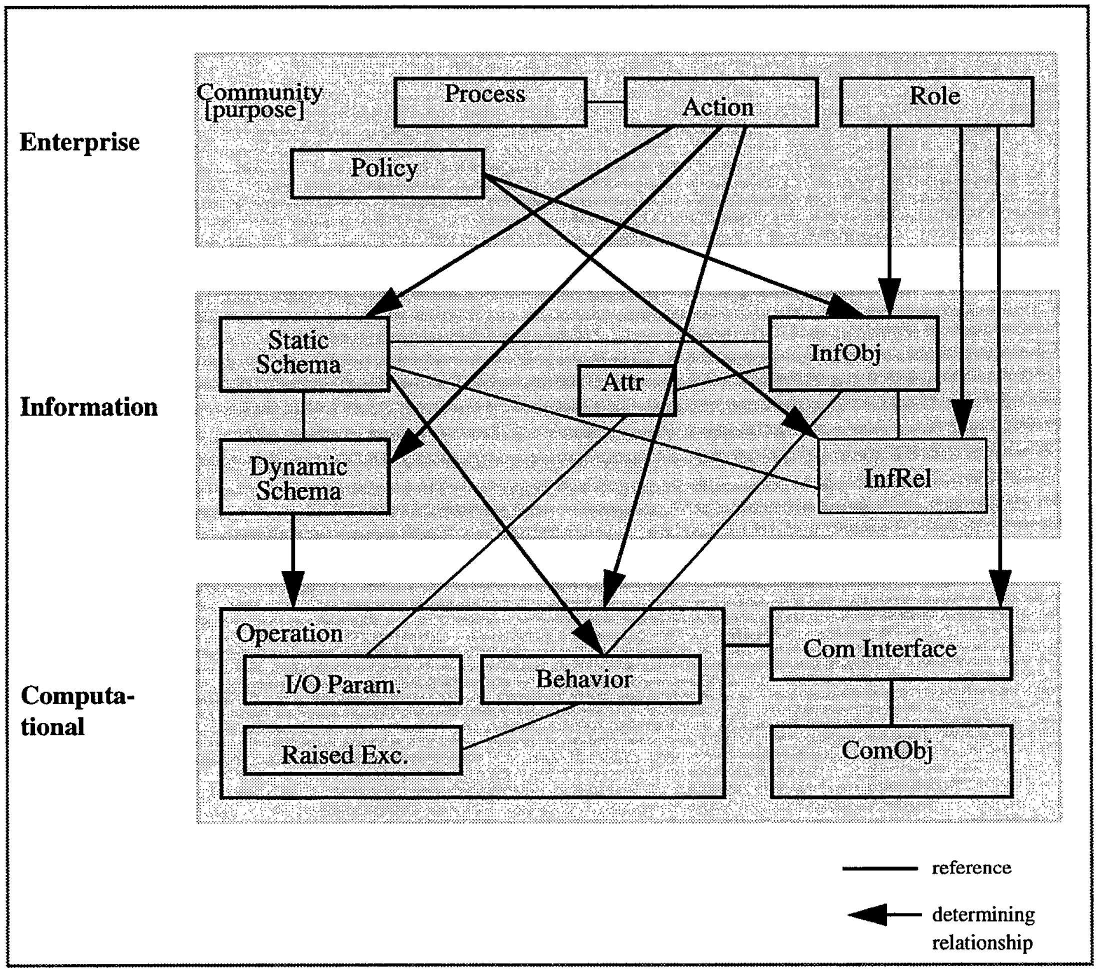



**Figure 8. Inter-Viewpoint Matching Approach**

## 4. DGSA and RM-ODP

In this chapter, we use RM-ODP viewpoints to introduce concepts from DGSA. The primary concern of the DGSA is security. Therefore, from the standpoint of constraint-based architectures, any constraint not relevant to security is omitted from our main consideration.

In Figure 9, the black ovals represent the security elements of each viewpoint specification of a system. The non-security elements are arbitrary, so long as all viewpoints are consistent at conformance points. In the following sections, we will examine each viewpoint in turn to see what features (constraints) remain.

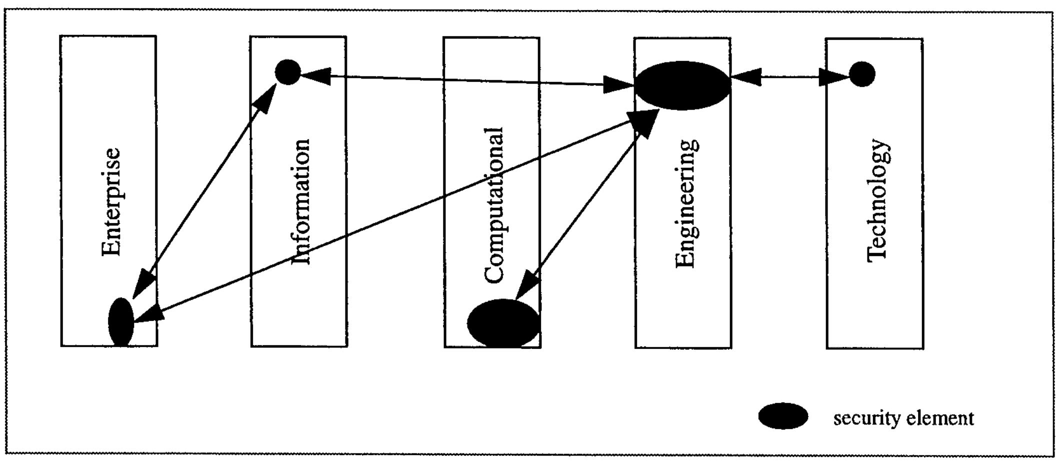



**Figure 9. Mapping of DGSA "Security Viewpoint" to RM-ODP**

### 4.1 DGSA Concepts from the Enterprise Viewpoint

The enterprise viewpoint represents an information system as a community, which is a configuration of objects formed to meet an objective. The community objectives that we consider in this report are those that facilitate sharing of information among a group of users while maintaining appropriate protection as specified by the security policy for the community. While the DGSA is vague on what a user is, we identify a user with a particular assignment of values to the security attributes, which is the granularity at which the security policy can distinguish among processing entities. Given the community objectives, we require that the actor roles of the community also be distinguished by this assignment to security

attributes. Systems will have at least as many communities as there are unique security policies and separate groups of actors, of which there may be a large number.

We impose these restrictions:

- That an actor role (a collection of behaviors) in a community must have the same obligations, permissions, and prohibitions to each object taking on the role of artifact in that community. All artifacts are shared in the same manner. Thus, if a particular actor role is permitted to modify one artifact in the community, it is permitted to modify all of them. This does not mean that all actors have the same permissions. A teller role might have modify permission in the community, while an auditor role only has observe permission.
- Uniform protection applicable to artifacts within a community represents an important simplification of DoD's MultiLevel Security policy because within a community we need no longer distinguish sensitivity of objects or their category while operating on them. Only when artifacts are exported from one community to another must policy regarding release be consulted. Further, no proofs are necessary that artifacts of a high sensitivity level in a community can become artifacts of low sensitivity level in that same community. All objects are at the same sensitivity level within a community. A proof regarding information flow between communities may still be required but is simplified by this fact.
- Finally, uniform-protection communities provide natural boundaries of a suitable granularity at which to assure that information flow is limited and that all objects within the community operate in accordance with the security policy.

Artifacts in different communities are considered to represent different pieces of information even if their values and roles are the same. Modifying an artifact in one community has no effect on artifacts in any other community. We only allow information to flow between communities using an explicit transfer, provided that this transfer is allowed by the security policy and that an actor role present in both communities initiates the flow.

A community formed according to these rules corresponds to a DGSA *information domain*, as shown in Figure 10 on page 33. The elements of an information domain are information objects (artifact objects and resources), users (actor objects), and a security policy (obligations, permissions, and prohibitions for each user).

The DGSA does not describe the class of security policies that might be implemented in a community other than by the requirements described above. It does specify that no implicit hierarchical trust or sensitivity relationships can be inferred between communities. All rela-

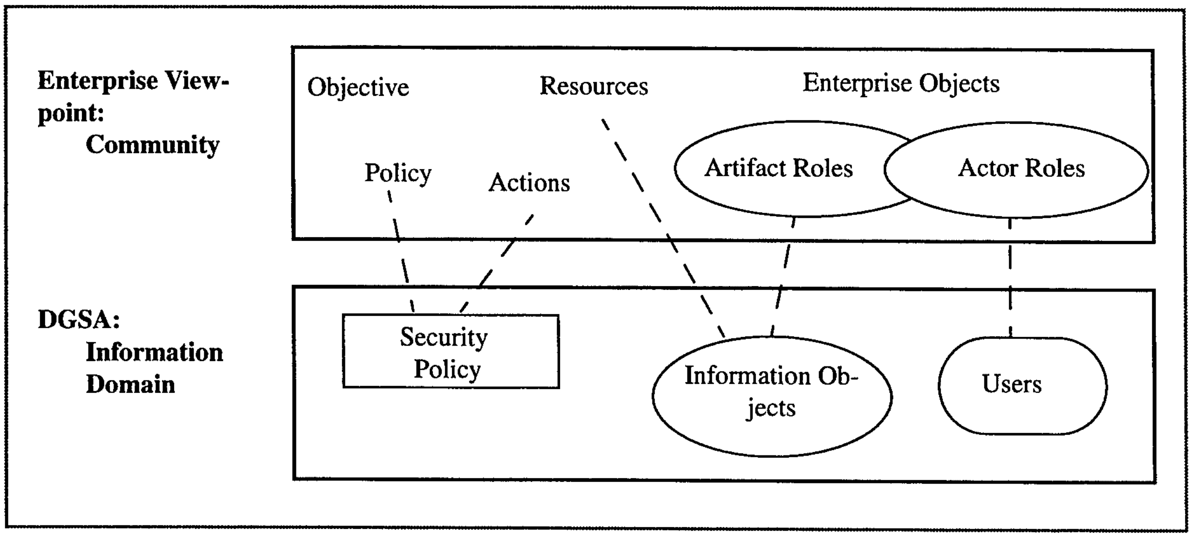



**Figure 10. Correspondence of Enterprise Viewpoint and a DGSA Information Domain**

tionships between communities must be specified explicitly through permissions to import from other communities or to export to them. This models the new DoD situation in which the United States is participant in many coalitions: information shared within one coalition may not be shared with another. This also models the general situation in commerce better than the hierarchical structures formerly required by DoD and described classically by the Bell-LaPadula Multics interpretation [Bell-La Padula], which only dealt with control of access to targets by principals.

An Enterprise object may itself be represented as a community at a lower level of abstraction—such an object is called a *Community Equivalent Object* (CEO). Thus, pieces of information that at a high level of abstraction might be in the same community may be refined into different communities. While the DGSA does not discuss the design process and multiple levels of abstraction, it has a notion similar to that of a CEO called a *multidomain information object*. These are *meta-objects* that are compositions of information objects from different information domains and are used to display and transport conglomerations of information. For example, some reports consist of paragraphs that are each labeled with their security classification. The multidomain information object maintains the configuration representing the relationship between the parts, much the way a community of CEOs does.

An example of such a composite community is a coalition organization whose objective is to issue an OPORD (Figure 11 on page 34). The actor roles are the coalition command group and each of the countries in the coalition, and an artifact role is the OPORD. The organization of the community represents the allowed lines of communication as specified in the rules for the coalition; for example, the countries may talk with the command group but not directly

with each other. This community can be refined by representing each country actor with a community in which the actor roles are the various units provided to the coalition by a country, and the artifact roles are that country's views of the OPORD, along with other pieces of information. Information may be transferred among the communities along the lines established in the high-level community.

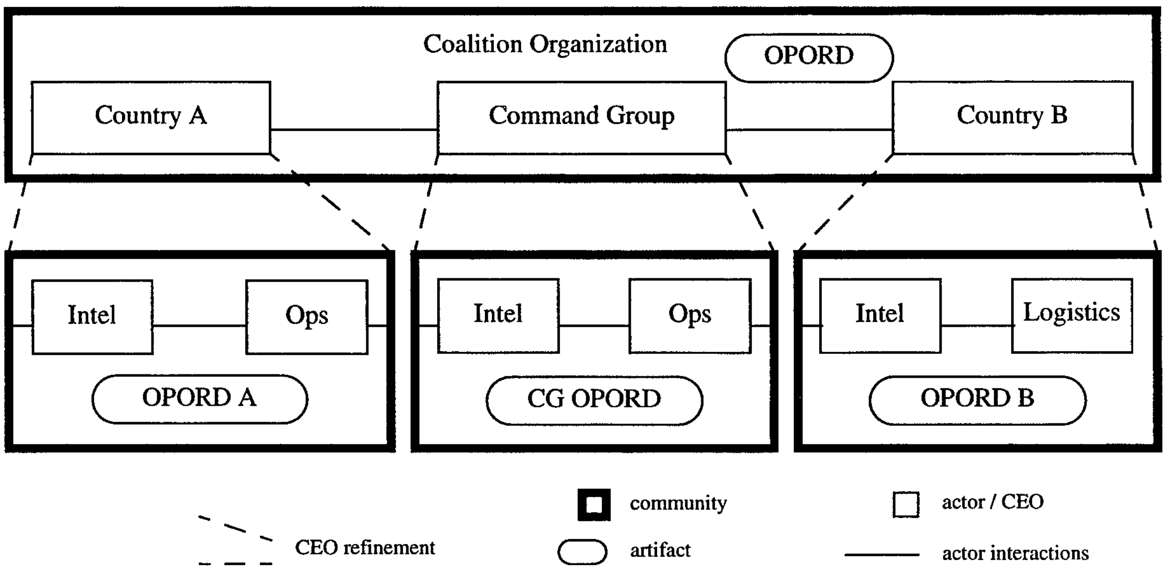



**Figure 11. Composite Community: Coalition Organization**

### 4.2 DGSA Concepts from the Information Viewpoint

The information viewpoint consists of information objects and static, dynamic, and invariant schemata that describe their states. Most of the information objects pertain to the mission for which the information system is used; these objects and the schemata defining the types and the relationships among their values depend on that mission. Some information objects and schemata pertain to the security of the mission information objects; these will be discussed in this section.

Security requirements in the information viewpoint are expressed with invariant schemata that restrict the static and the dynamic schemata so that they can only represent secure states and state changes. These requirements must correspond to the policy specified in the enterprise viewpoint. Thus, if an action is prohibited in the enterprise viewpoint, the corresponding dynamic schema must be prohibited by the security invariant schemata.

We are interested in dynamic security policies in which the set of allowable state changes might change over time. The security invariant schemata therefore must be parameterized to correspond to the current policy. One way to do this is to provide the security requirements as

a parameter, which emulates the DGSA requirement for the separation of decision about and enforcement of security policy. Thus, the security requirements are represented as a set of information objects. Here the ISO model of decision and enforcement for access control is of special note [ISO10181-3]. Although this model applies specifically to access control, it may be extended to all elements of the security framework. It relates decision to security information that is obtained from the initiator, the target, the requested operation, the security policy, and the "information retained from previous decisions" as well as the current state of the system (e.g., time, date, condition (normal, under attack, etc.)). The model permits policies that depend on the subject's previous actions or are conditioned on current system attributes, e.g., wall clock time.

The DGSA calls the set of objects specifying the security requirements for an information domain a *Security Management Information Base* (SMIB). The information viewpoint can represent a SMIB as a static schema *smib* that maps each information object to the collection of information objects representing its information domain's SMIB. Note that an object in a SMIB is itself an information object and is therefore mapped by *smib.* A SMIB must contain a information about the users that may access the information domain, and for each user a set of security attribute values that may be used to determine permissions, prohibitions, and obligations. In Figure 12, the arrows represent *smib*, mapping each information object to its SMIB. Objects a, b, and c contain mission-related information while objects d, e, f, and g contain security-related information. The SMIB for a and b is d, the SMIB for d is g, the SMIB for c is the pair of objects e and f, and g is its own SMIB.

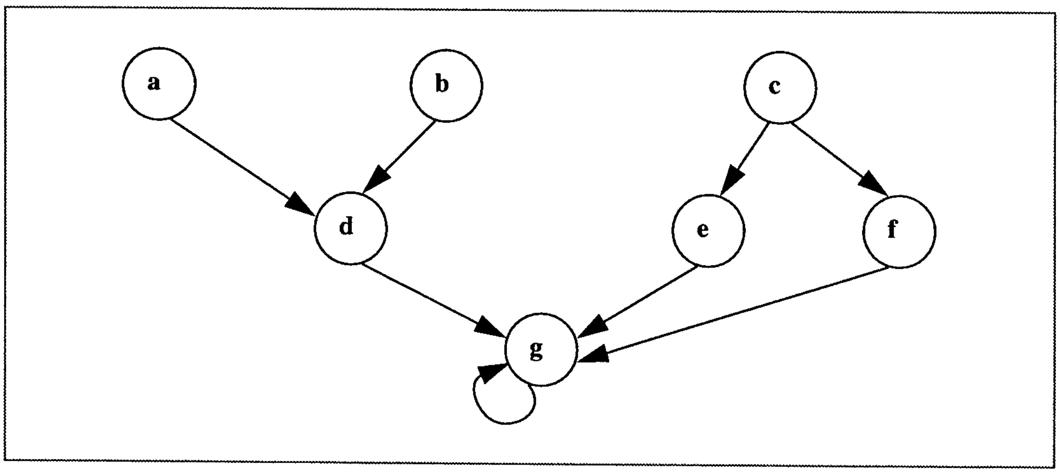



**Figure 12. A *smib* Static Schema**

All information objects related to artifacts in a single enterprise community must utilize the information from the same SMIB since they all are subject to the same policy; this SMIB must be an encoding of the community policy. Figure 12 represents at least three communi-

ties: {a,b}, {c}, and {d,e,f,g}. A dynamic schema that creates a new SMIB therefore corresponds to the creation of an enterprise community.

A multidomain information object, described in the enterprise viewpoint, can be represented in the information viewpoint using a static schema. This schema relates a meta information object representing the multidomain object to the information objects from which it is composed. The *smib* schema relates the meta-object with a SMIB that specifies which users may alter the composition of the object and which users can display the object.

In the DGSA, most operations occur within an information domain. An invariant schema that enforces this is that a static schemata may include objects from multiple domains (objects that *smib* maps to different SMIBs) only if it either is *smib* itself or represents a multidomain information object. Similarly, a dynamic schema may include objects from multiple domains only if it modifies either a mapping in *smib* or a mapping between a multidomain object and its constituent parts. A dynamic schema that changes the *smib* mapping for some information object changes the community of the corresponding enterprise object and therefore corresponds to an information transfer, which is subject to the security policies as specified by the old and the new SMIB. In the sequel, we will often refer to a dynamic schema that changes either the *smib* mapping or a multidomain object mapping as a *meta-schema*.

For the OPORD system, the command staff and each country maintain a set of SMIBs. Any information possessed by one of these organizations, including its SMIBs, is mapped by *smib* to one of that organization's SMIBs. Users that do not have authority to create policy for an organization are prohibited from modifying that organization's SMIBs.

### 4.3 DGSA Concepts from the Computational Viewpoint

The computational viewpoint takes an object-based approach in which all processing is encapsulated within objects. The DGSA requires *strict isolation* between information domains, which means that information in one domain can only affect processing in another domain through an explicit transfer between the domains as allowed by the security policy for those domains. Therefore, all other objects with which a computational object interacts must be associated with the same enterprise viewpoint community—as is that object. Thus, computational objects form interacting clusters that match the communities. The only interaction between clusters is the detachment of an object from one cluster and the attachment to another, corresponding to the transfer of an enterprise object between communities.

Figure 13 on page 37 shows an example computational viewpoint of a system. The RM-ODP communities give rise to a system that support three distinct information processing

domains, which in turn correspond to three information domains. The diagram shows that both A and B may transfer objects to C; similarly, C may transfer objects to both A and B. Otherwise computation is limited to computations within A, B, or C. Of course the number of objects and their configuration need not be limited to four objects or simple configurations as in each of the information domains pictured. We note that in this viewpoint there is no notion that the objects reside at a specific site in the network: the information objects corresponding to the computational objects may be distributed to any collection of distributed systems that will support the information domains and their policies adequately.

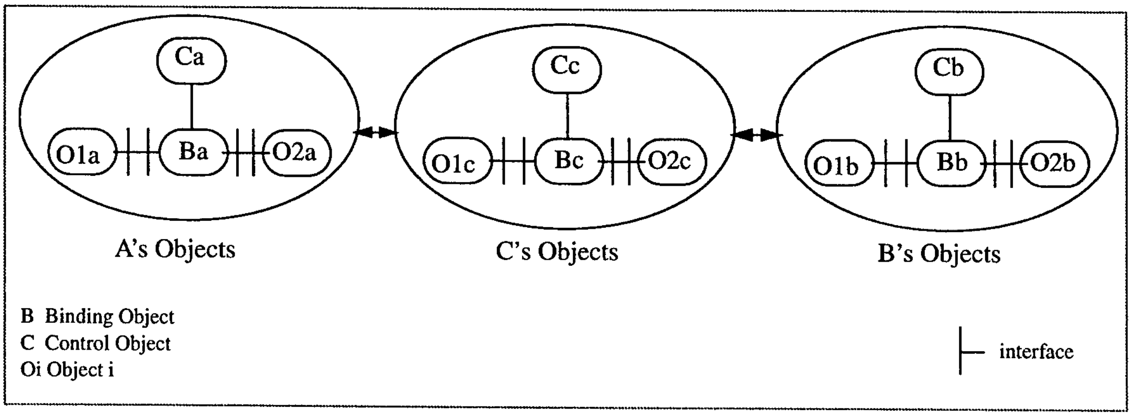



**Figure 13. Computational Viewpoint and Information Domains**

Each computational activity within an object must be identified with a principal in order to limit the allowed actions according to the security policy. The SMIB of the information viewpoint for the information domain of the object must identify the principal as a member of that domain. Some objects are directly linked to a principal and all computational activities within that object belong to that principal. Other objects can process on behalf of multiple principals, and delegation rules for the invocation of processing are used to determine the principal for each thread of computation.

### 4.4 DGSA Concepts from the Engineering Viewpoint

Figure 14 on page 38 shows the engineering viewpoint of a DGSA system similar to that of Figure 13, but is somewhat simplified in that all information domains are hosted on one machine. Correspondences between the figures are:

E1:O1a E2:O2a E3:O1c E4:O2c E5:O1b E6:O2b

Because E1 and E2, E3 and E4, and E5 and E6 happen to be located in the same capsule, no binder is required for operation. Management of the binding occurs in the cluster manager in

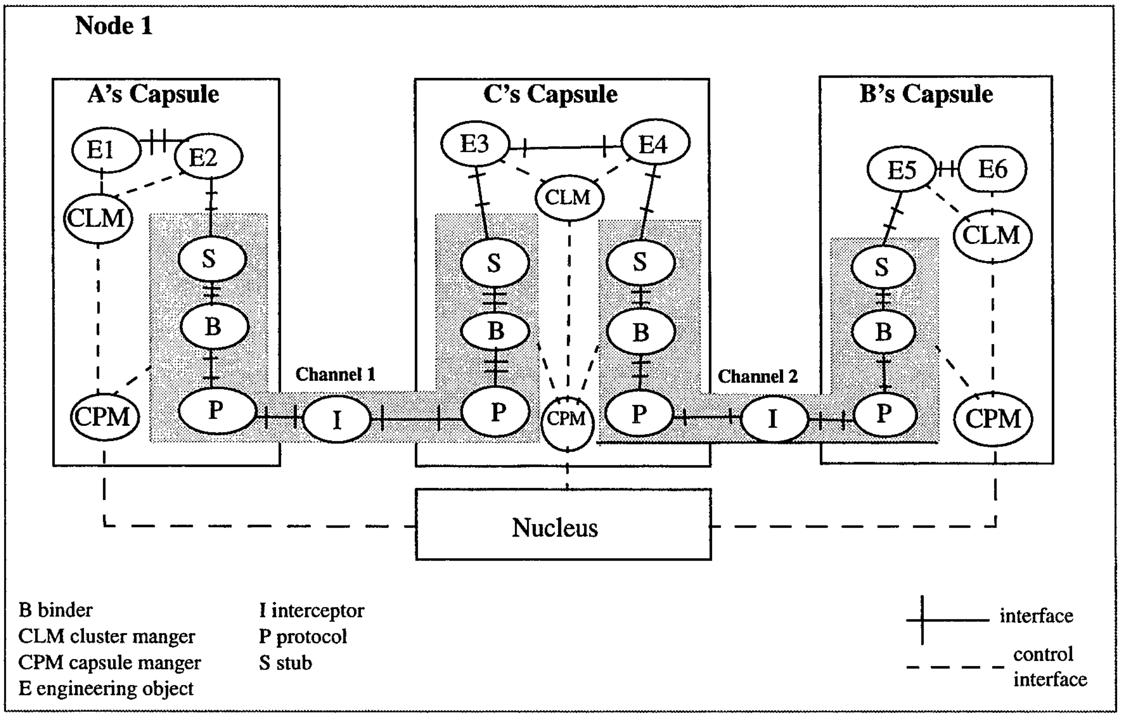



**Figure 14. Engineering Viewpoint**

the A, B, and C capsules. The capsule managers control the channels that permit the different information domains to exchange information.

Key allocations of DGSA security functionality are given in Figure 14.

- Isolation is a principle requirement.
- The capsule is used to isolate all information objects in an information domain on a given node from all other information objects on that node.
- A node is used to isolate all information objects in information domains on it from all different information domains on other nodes.
- The single- and multi-point channel provides information flow to implementations of the same information domain on other nodes. It must, in conjunction with the capsule, prevent the transfer of information between itself and other channels. A channel may provide information flows between different information domains in different capsules on the same node in a manner governed by the meta-security policy.

The stub object is responsible for composing information to be transmitted over the channel into messages and then decomposing messages received. The allocation of security func-

tionality to stubs might include privacy-related functions. When the channel is between nodes, the protocol object will usually encrypt messages that it is sending and decrypt messages that it receives. Usually the stub object will not use encryption when the channel is contained on a single node (usually implemented by the nucleus using interprocess communication or shared memory).

The binding object (of the channel) manages end-to-end integrity of the channel. The allocation of security functionality to it may include integrity on incoming and outgoing messages. The stub object may also be allocated functionality relating to the availability of the channel.

The protocol object provides additional data to the channel using the channel protocol. The allocation of security functionality might include identity and authentication functions. For example, it may provide/verify identity and authentication information to/from the other party(ies) using the channel. A particular concern is the translation of principal identities if the capsules use different identifiers for the same principal.

DGSA requires that the security enforcement function is performed separately from the decision function.[1] This separation presents the possibility that a different style of enforcement will be used to implement the policy on different computing platforms or in different domains. Both platforms or domains will enforce the decision but possibly in different ways with differing combinations of equivalent security enforcement mechanisms.

### 4.4.1 Interdomain Transfer of Objects

When multiple information domains are represented on an end system, the end system must provide strict isolation between the domains as described previously. Each information domain must be encapsulated as though running on a virtual machine such that virtual machines cannot obtain any information from or about each other. If the policy explicitly permits, a channel can be established from one virtual machine to another. This channel must be confined to an end system to permit adequate control and monitoring of the channel. Also, information in the channel must be tagged with the principal that sent it; the same principal operating in the receiving virtual machine must accept it.

When an object is transferred to a different information domain, it acquires the security attributes and their values for the new domain. For example, every information object might have an identifier that is unique within its domain; a different type of identifier might be used

[1] This permits the notion of precalculation of rights and caching of decision information (if permitted by the security policy), which can provide significant performance gains through the use of cached rights [DTOS].

in the new domain. Setting the security attributes is done by the interceptor object shown in Figure 14 on page 38. This setting must be atomic so that an object is in one domain or the other and never simultaneously in both domains. Note that the object itself need not actually be moved in memory or modified.

### 4.4.2 Security Associations

An information domain may extend beyond the bounds of an end system to encompass a group of systems that "share" its information objects. Each of the encapsulated security contexts, or *sited domains* [S99], that make up an information domain must communicate with each other with the same assurance as if they were on the same end system. A *security association* is the totality of the communication and security mechanisms and functions that securely bind together sited domains of an information domain [DGSA]. While each of the end systems is expected to properly enforce the domain security policy, this is not a property that a security association can enforce but rather is a trust relationship between end systems. In Figure 14 on page 38, the interceptor objects (*I*) provide the security association. Because an information domain may be on many end systems, a multipoint channel with multiple interceptors may be required.

A security association must transfer security policy, information objects, and threads of control while maintaining strict isolation between the information domain that it supports and all other communications. It is used to establish agreement on communication protocols, including data formats, and cryptographic protocols and keys. Also, any changes to the set of principals in the domain or to the domain policy must be conveyed to all of the end systems in such a manner that the policy remains consistent. Note that the security association is a function of the user attributes, platforms involved, and the overarching security policy. A different security association is used for each information domain.

The DGSA requires that an individual information object exist on only one end system at a time, but the domain security policy may permit copies.[2] This means that the task of maintaining consistency of replicated objects is left to the information domains. A more natural solution would have been to include object consistency rules as part of an information domain definition and to control replication as part of the security association.

2 The copies may be used to enhance reliability, efficiency, or both.

### 4.5 DGSA Concepts from the Technology Viewpoint

The technology viewpoint deals with RM-ODP standards and products. Different domains may use different security products. For example, different hosts may use different processors and/or operating systems that will have different security characteristics relevant to the implementation of security functionality on that system. The implementation of channels using conventional transports may utilize security standards such as the Internet Protocol Security (IPSEC) Internet Engineering Task Force (IETF) Standard.

Assuming a homogeneous technology domain, a sample technology viewpoint can be found in Figure 15. In a heterogeneous technology domain, components that support an information domain may require some form of converter. In this viewpoint all standards, components, and technologies used in the system that are relevant to security should be listed.

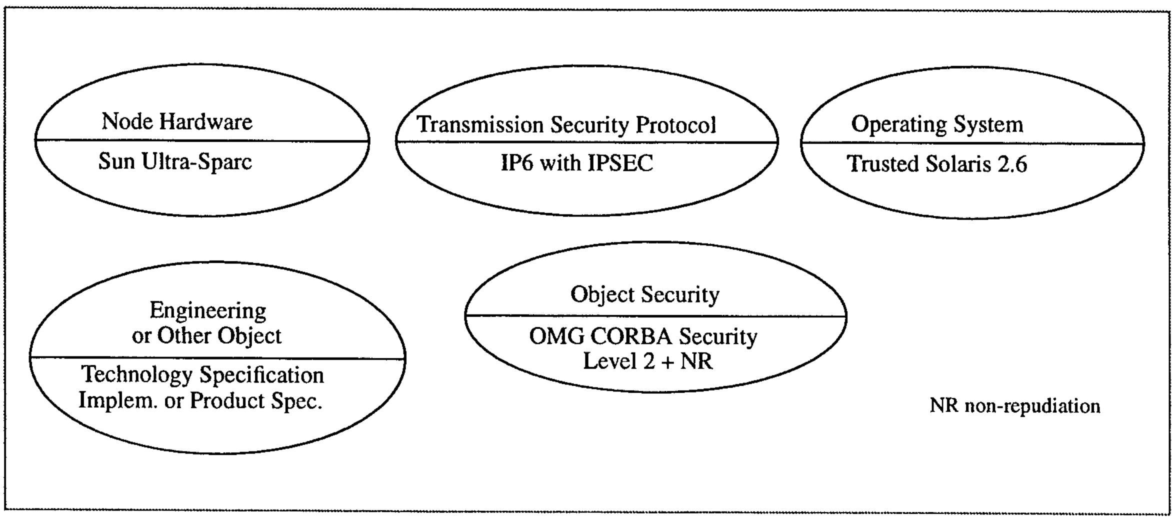



**Figure 15. Technology Viewpoint**

### 4.6 Implementation Issues

We have shown the different views through which a system and its security policy must be considered. The system designer must be able to move from one view to another. In this section, we discuss some of the details and problems that can occur.

#### 4.6.1 Assessment of Proposed Operation

Because no computational or engineering models are prescribed to the implementor by the DGSA, generic models are needed. The information domain concept is to be implemented on a wide variety of platforms, each of which may have an optimal mechanism for enforcement of one aspect of the security policy. To develop such a model, an assessment of proposed oper-

ations document must be generated that details the way in which the system is to be employed. The security portion of this document would detail the security risks to the system, describe any expected adversaries, and categorize the value of the assets to be protected. It would also outline the security policy to be enforced and the security objectives to be met by the system. Examples of these procedures may be found in the Common Criteria [CC]. This is the key to the generation of an optimized design for an implementation.

Depending on the class of security policies to be implemented, the design of the system may be simple or hard. For example, if the security policy does not have specific integrity or availability requirements, and need not deal with distribution, a virtual machine model may be sufficient. In this model, each information domain is implemented by a different virtual machine. Information domains may be linked by means of virtual machines sharing a private communication channel. If the policy restricts information flow, the implementor may focus his attention on the boundaries of the information domain and need not concern himself with information flow within the information domain. This simplification of concept correspondingly simplifies the implementation on a variety of machine types and can greatly raise the level of assurance that the policy is being correctly enforced.

Access control is not the only concern of security policies in the DoD. Examples of desired security policies can easily be related to the ISO/IEC Security Frameworks for Open Systems[3] and several are given in Table 7 on page 43. More complicated policies, involving multiple coalitions, will likely be required in actual DoD missions or in employment by industrial consortia. The following tables briefly assert that a DGSA-compliant architecture may be required to realize a wide variety of access control and integrity polices.[4] Policies dealing with availability and accountability have not been categorized well enough to permit such a list.

### 4.6.2 Principles

Before considering the implementation, it is worthwhile reviewing classes of security policies that have been suggested in the literature. One of the better papers on categorization for access control is by Saltzer and Schroeder [SS75]. They divided the set of access control policies into five classes based on the amount of sharing to be done. Table 7 summarizes this classification and how it affects the DGSA implementation.

---

3 ISO/IEC 10181, parts one to seven (see the list of references provided in this report).

4 This is not to imply that the implementations would be simple, straightforward, or efficient in all cases.

**Table 7. Functional Access Control Levels**

| Policy Type for Control | Affect on DGSA |
|---|---|
| Unprotected | One information domain: PUBLIC; any user is accepted. |
| All-or-Nothing (also known as System High) | One information domain: PRIVATE; only authenticated, authorized users are accepted. |
| Controlled Sharing | Different access rights on each object require multiple information domains—at least one for each kind of security policy with additional domains based on distinct sets of users per domain. ACLs are not required, but copies of objects in different information domains must be synchronized. |
| User-Programmed Sharing Controls | Protected objects and subsystems reside in separate information domains with a domain per content type or per security policy (based on a different context). |
| Labeling Information | One domain per distinct label; label need not be explicit. Label relates to import and export policy. |

Source: [SS75]

One of the few papers on integrity and integrity policies, published by NSA [NSA91], provides the information presented in Table 8. It divides integrity into four classes of policy. This table indicates that DGSA principles can be used to realize a wide variety of confidentiality and integrity polices. Availability policies have not been as well categorized to permit such a list, and require additional study.

**Table 8. Integrity Policies**

| Policy | Sub-Policy | Affect on DGSA |
|---|---|---|
| Identification and Authentication | User I&A | Information domain membership policy. |
| | Originating Device | Originating device treated as an information domain. |
| | Object | Originating object treated as belonging to an information domain. |
| Authorized Actions | Conditional Authorization | "State" of the information domain provided for decision making. |
| | Separation of Duties | Security attributes providing role or task information and context information used for decision making. |
| Separation of Resources | Address Space Separation | Information domain. |
| | Encapsulation | Information domain. |
| | Access Control | Information domain and decision making module. |
| Fault Tolerance | Summary Integrity Checks | "State" of an information domain. |
| | Error Correction | "State" of an information domain. |
| Source: [NSA91] | | |

Part of the difficulty of discussing the implementation of such a wide range of policies is that the DGSA does not provide implementation guidance. It does not say how information objects are to be segregated. For example, it does not require that information objects must be explicitly labeled. Information objects produced by combining values from other information objects must either be "labeled" implicitly or explicitly in the sense that the label identifies the information domain (or community) with which they are associated. Should they be copied into new information objects, combined with old objects to produce a new object, or exported to other information domains, their labels can be used for enforcement of the security policies that govern them.

If integrity or audit policies for objects within an information domain are to be implemented, it may be profitable to think of the information objects as existing within a software architecture such as a message-oriented Object Framework. In this architecture a trusted agent receives a message from a user attempting to perform a mediated operation on an information object. The security decision is whether or not to carry out the operation. If permitted, the agent carries out the mediated operation utilizing trusted code. The implementor must provide a mediator that intercepts requests for operations that are performed on information objects and are controlled by the security policy. He must assure that every operation is performed (if allowed) as required by policy. One way that this could be achieved would be to have principals housed in one capsule and objects in another with a channel between the two capsules. The security mediator would decide whether any requested operation was to be performed, and would perform the operation itself, returning results via a different channel. Such a structure might be particularly useful in maintaining integrity of a collection of information objects.

If several domains with similar but distinct security policies are represented on a given machine, plugable security modules in the manner of IPSEC can be used. In this event, table-driven security policies can be used to select the level of assurance by determining such parameters as the strength and style of identification and authorization, access control, privacy, integrity, and auditing. The table entries would select the appropriate mechanism and parametrically determine the version of the security service to be used. With care, a wide range of policies might be accommodated with a modest selection of types of service and levels of service that are implemented.

For example, some of the application programming interfaces (APIs) that might be used include the Microsoft Cryptographic API (CAPI), The Open Group's[5] General Security Ser-

[5] Formerly X/Open and Open Software Foundation.

vice API (GSS-API), and the Intel Common Data Security Architecture (CSDA). These APIs support generic calls of security functionality. The underlying mechanism can be selected by the policy in force in the domain from which the call is made. In this way, the user would be able to use the same interface to the services without knowing which plugable module was actually invoked.

If distribution is to be allowed, the virtual machines may be linked by means of a security association that builds a virtual secure network from a virtual private network, perhaps using components of the IETF IPSEC suite of specifications. Use of this suite could enhance service interoperability. However, an Enterprise-level policy requires substantial refinement including management functions to be expressed at the technology level using IPSEC. This method also allows the use of COTS-based CORBA implementations without requiring that any CORBA service within a domain be secure or trusted.

### 4.6.3 Implementation of Multidomain Information Objects

The DGSA is silent on how multidomain information objects are to be implemented. For example, it does not indicate how the printed or displayed objects are to be visually labeled so that the human reader may be alerted to the security policy controlling their use. Clearly, each information domain needs a unique human-readable label that can alert the user that a security policy is in force. More information on security policies dealing with such composites is needed in order to determine a sensible implementation.

If an appropriate solution to the management of the output of these objects can be obtained, we can design their implementation. One method that may prove appropriate is to implement a scripting language with data structures consisting of opaque references. The procedure describing what is to be done with the information objects in scripting language can be imported into a domain to an agent representing the principal. If the security policy allows, the action can be performed and success or failure can be reported by exporting that information. Note that the information domain's policy must specifically permit use of its opaque pointers by means of the importation of the script and its exportation of the secure object handles.

There must be a security policy that describes the generation, storage, and use of each class of these meta-objects. Management of these multi-domain policies is likely to be complicated. It may prove so difficult to administer them that a more straightforward method will be employed to accomplish the desired end: the establishment of a new domain to which the elements of the object may be transferred from each of their constituent domains. The new domain would have its own human-oriented label and its own policy regarding output. A key issue is the formulation of the security policy for the resultant domain.

### 4.6.4 Management

Management of the information domains is a key issue for the implementor and the user. This is because of the variety of security policies that are feasible and because this variety may cause users to wish to have a large number of information domains to precisely control information sharing. The critical issue is that the cost of ownership not be dominated by the cost of security management. Information domains must be easy to create, modify, and destroy. Further, certifying that they support their security policy (even in the face of distribution of information objects) and gathering the information that leads to accreditation (so that they may be used for the purpose intended) must be easy.

As discussed in Section 4.4, the DGSA leaves management of replicas to the information domains. This makes it difficult to link availability requirements in the security policy with the services provided by an end system. Each end system needs to provide a set of coherence protocols from which a selection can be made for each domain. These protocols also order actions from which an audit sequence can be generated.

Security policy expression and management will require extreme ingenuity and inventiveness in order to properly refine "natural language expression" of high-level security policy into covering requirements at successively lower levels so that the lowest level requirements yield actions based on security attributes and values. The same level of inventiveness will be required to properly design a collection of plugable security modules that implement the policy. The issue of who may change the security policy, how it may be changed, and what change means in a distributed system are still subjects of research. However, it may be hypothesized that security policy may be organized so that some or all attribute values may be changed dynamically, but the structure of a policy and its attributes may not be changed except through administrative creation/deletion of a policy.

Audit plays a big part in high assurance systems. Implementation must be done so that the audit process and its performance will not lead to disabling the audit. In addition, the implementation must be designed in such a way that the level of detail of the audit can be varied appropriately, perhaps on the basis of current perceived threat. Further, it is likely that real-time analysis of the audit may be required in order to permit real-time intrusion detection.

Because every object in an information domain has the same security properties, the domain identity is the only security attribute that is needed. When this identity can be determined implicitly, COTS and GOTS software may be able to be run without modification (depending on the security policy and provided it is run in a single domain, strictly isolated from other domains).

## 5. Coalition Example

We will use an OPLAN of a Coalition Task Force (CTF) as the example of our application of RM-ODP and DGSA. The purpose of this example is as follows:

- To show the mapping between the DGSA entities and concepts and the RM-ODP entities and concepts; and
- To present the constraints that must be imposed on RM-ODP compliant abstract architectures because of DGSA's requirements and constraints as the model for security.

We will limit our example in scope and detail to a small subset of the organizations involved in a real OPLAN, but a subset large enough to illustrate DGSA and RM-ODP. Even for this example, the detail is quite voluminous.

- A federation of three allied nations $N_1$, $N_2$, and $N_3$ wishes to work out a coalition OPLAN.
- Each of the allies will have several groups working in at least two levels of classification. We will focus our attention on the release of information from HIGH classification to LOW classification and from nation to coalition (and vice versa).
- Each nation $N_i$ has an intelligence group[1] $I_i$, an operations group $O_i$, a logistics group $L_i$, and a communications group $C_i$.[2]
- Each nation wishes to share information with other nations bi-laterally and tri-laterally.

At a high level of refinement, the enterprise viewpoint for this example consists of three communities as shown in Figure 16 on page 48.

---

1 Typically intelligence operations are regarded as having some aspects more sensitive than physical operations and deal with information considered to be at a higher classification.

2 While we could treat each of the organizations separately, i.e., have a separate information pool for each, in order to simplify the example, we will assume that all the organizations from a given nation at a given level of classification use the same information pool. The high classification pools will be limited to those in the respective intelligence groups.

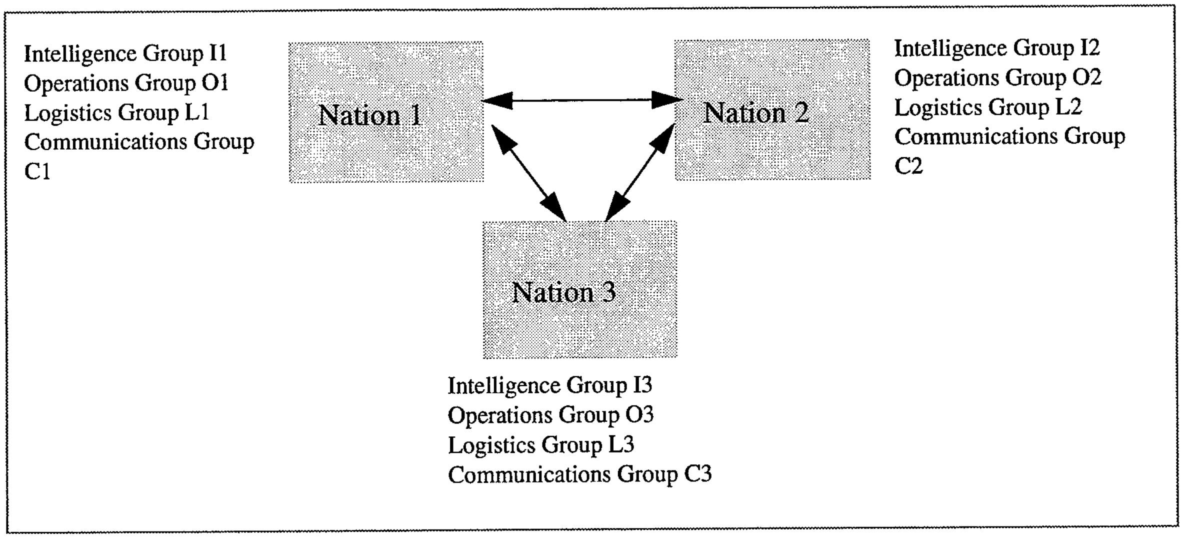



**Figure 16. Coalition Task Force**

Potentially, each group has its own shared collection of information (denoted *information pool*) for each information security policy that it supports.[3] Suppose that each group has information that may be read, written, or updated. For a particular nation, if several of the groups employ the same security policy for each level of sensitivity and are willing to have their combined membership share all information at each level of sensitivity for which individuals are entitled, their information pools can be combined. This is often the case and results in a few "system high" environments per nation. In some cases, however, the groups cannot or do not wish to have information pools shared with others and the statement of the total community use becomes significantly more complicated, e.g., the communications group may not wish to share its information with the logistics group.

For this example, we assume that the CTF will be a small operation and that each nation will use two "pools" of information because of different security policies or memberships, one classified at a lower level (LOW) and one at a higher level (HIGH).[4] These pools will be denoted N1L, N1H, N2L, N2H, N3L and N3H. Four additional pools of information denoted N1N2L, N1N3L, N2N3L, and N1N2N3L will be shared by various combinations of nations from the coalitions, for a total of 10 shared information pools. Each of the four is an information pool to which all cleared users from more than one nation will have access. N1N2N3L will contain the final "general" plan and is marked at the lower level of joint classification. In

3 This would yield 56 information pools—4 groups, 7 combinations of nations, 2 security levels.

4 The HIGH and LOW sensitivity levels of each nation are incommensurate with those of the other two nations. The LOW sensitivity level for each of the joint pools is incommensurate with the LOW sensitivity level of each nation.

keeping with the application of the policy of *least privilege*, the groups using all information pools will be divided into two groups: the observers and the participants. Observers are allowed to look at all information but not change any; participants are permitted to change the contents of the plan.

A key to successful operation of the enterprise is the sharing of some of the information used by a subcommunity with other subcommunities. It is extremely important that the federation of information systems that support these communities be able to accomplish this in an automated fashion to achieve accuracy of the transferred information and timely delivery. If possible, the automation should directly implement the technical security policies of the communities exchanging information. The rest of this chapter will focus attention on the constraints and abstractions required to do this.

Another key to development of successful automation is the choice of a security policy or collection of security policies that permit the desired automation to be accomplished.

Each information pool and the actors that may access it forms an information domain. The DGSA defines two invariants on the policies of these domains:

> **Policy invariant 1: A permission.** Use of subcommunity information is permitted to all actors who are members of communities sharing the information pool, but only as permitted by the security policy governing the pool.
>
> **Policy invariant 2: A prohibition.** Exportation of information from one shared pool of information to another shared pool of information is prohibited unless specifically permitted by the security policies governing the pools and unless the export of information is performed by an actor who is a member of both the exporting subcommunity and the importing subcommunity and who has a role specified by the security policy that permits exportation.

Other requirements from the DGSA relating to technical security policy will be elaborated in the viewpoint in which they are seen.

In our example, the technical security policy[5] governing transfers of information between communities of the coalition is based on the Wiemer-Murray Domain Security Policy Model for International Interoperability [WM98]. This policy will be regarded as a "meta-security policy" because it does not govern subcommunity use of information but instead governs intercommunity information flow.

5 As a subset of the complete organizational security policy.

### 5.1 Wiemer-Murray Domain Security Policy Model

The Wiemer-Murray domain security policy is an information flow policy typical of what might be used for an international coalition. In Wiemer's example, the coalition consists of Australia, Canada, New Zealand, United Kingdom, and United States of America [WM98]. Their policy, much of which is quoted verbatim and with some material paraphrased, is presented.

#### 5.1.1 Environmental Assumptions

Three security attributes characterize information domains in this policy, yielding five information domains in the example given in Figure 17:

- Classification label (CL*n*). Figure 17 shows a single classification: CL1. An example classification label is SECRET.
- Caveat label (C*n*). Figure 17 shows five caveat labels: C1, C2, C3, C4, C5. Caveats typically label a group that can access the data, e.g., NATO, CANUSUK.
- Operational label (OP*n*). Figure 17 shows three operational labels: OP1, OP2, and OP3. An operational label typically gives the name of the nation(s) in operational control, e.g., US, CANUS, UKUS

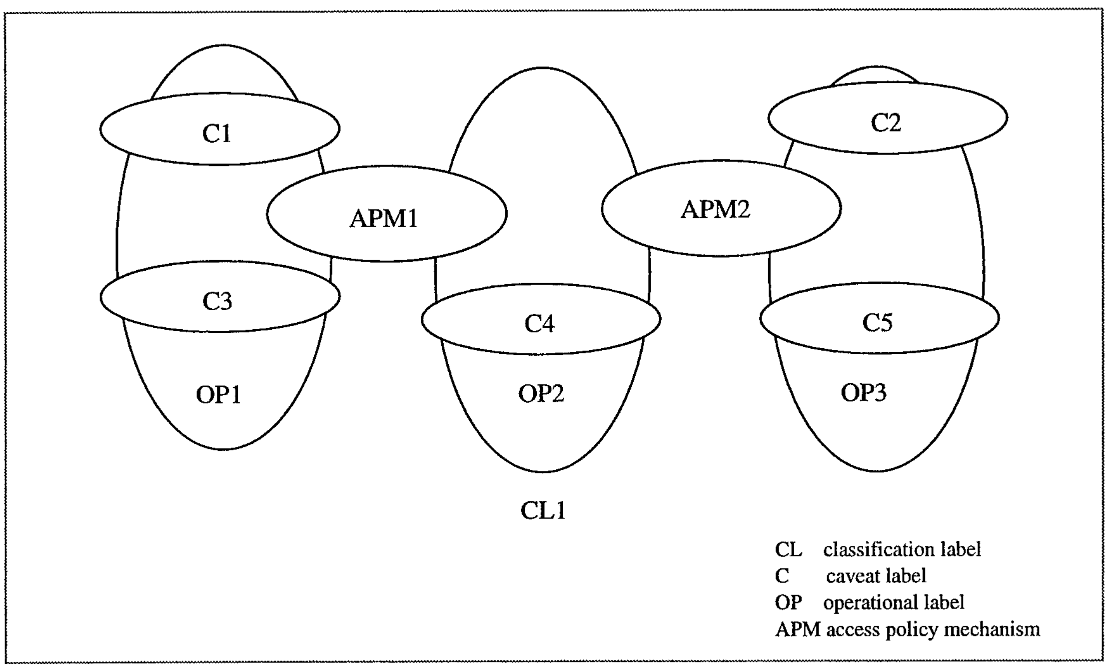



**Figure 17. Model Framework**

The security attribute labels may be implicit (based on the physical attributes of the connection) or explicit (based on binding of a logical label). An Access Policy Mechanism

(APM) is available to mediate the initiation of access between information domains. Figure 17 shows two such mediations (each of which could be different): APM1 and APM2. A transfer policy is available to mediate the transfer of information between information domains.

### 5.1.2 General Policy Model

To be a member of a domain, the user must satisfy the following three policy statements:

- Clearance policy: The user must have the requisite personnel security clearance for access to the classification of the domain, as defined by the classification label.
- Need-to-know policy: The user must have a recognized requirement to have access to the information.
- Formal access approval policy: The user's nationality must coincide with caveat of the domain, as defined by the caveat label. Also, where the user is to be a "release authority," the user must formally agree to abide by the transfer policy (discussed later in Section 5.1.3).

A connection may be initiated provided that the initiation of a connection is uni-directional and must be initiated from a more restrictive to an equivalent or less restrictive domain. (Note that once the connection is established, communications may be bi-directional based on the transfer policy in Section 5.1.3.) The following definitions apply:

- *More restrictive domain* is defined as one of two information domains in which the protective mechanisms associated with the characteristic labels of that domain imply greater protective mechanisms and the domains are *not mutually exclusive.*
- *Less restrictive domain* is defined as one of two information domains in which the protective mechanisms associated with the characteristic labels of that domain imply lesser protective mechanisms than of the other, and the domains are *not mutually exclusive.*
- *Equivalent domains* are defined as two information domains that share exactly the same set of characteristic labels.
- *Mutually exclusive domains* are defined as two information domains that are neither equivalent, more restrictive, or less restrictive (i.e., NATO and AUSCAN-NZUKUS are mutually exclusive information domains).

### 5.1.3 Transfer Policy Model

For the purposes of the Wiemer-Murray Model, the transfer policy is used to determine if the receiving domain has the appropriate protective mechanisms for the information to be transferred.

A user, having met the policy requirements to be a member of both information domains and having been granted rights-of-release authority, may transfer information between domains provided that:

- Users in the receiving domain meet the membership policies (clearance, need to know, and formal access approval).
- The information is verified as releasable.
- The transfer of information is controlled from a *more restrictive* or an *equivalent* domain. Users in a less restrictive domain may never gain control of information transfer.
- The release authority is responsible to ensure that information does not have a type of "eyes-only" caveat not suitable for the receiving domain, and that the information is of a suitable classification for the receiving domain.

## 5.2 DGSA and the Wiemer-Murray Policy

The DGSA does not predetermine characteristics of persons becoming users in a domain, but merely states that the user community determines who is to be admitted as a user. Wiemer provides a test that limits who may be a user of the domain based on clearance, need-to-know, and formal access approval. He also indicates that a user who is granted "release authority" must "formally agree to abide by the Transfer Policy." We assume that all persons in the CTF join all information domains that they are eligible to join, given their nationality, clearance, and their need to know; and that all persons authorized as release authorities have formally agree to abide by the transfer policy.

The DGSA does not predetermine a relationship between the policies of information domains—that is, whether they are more restrictive, less restrictive, equally restrictive, or incommensurate. The Wiemer-Murray policy defines a set of classification label rules, caveat label rules, and operational label rules, and a set of relationships based on them that are associated with the information domains representing the information pools described previously. The relationships are defined in order to provide a policy that corresponds to current coalition usage. In Section 3.6 of [WM98], the author demonstrates how the Wiemer-Murray policy (in a different but comparable example) complies with the definition of the DGSA.

The DGSA does not specify the detailed process of establishing communications between domains. This set of rules [WM98 pp. 534-5] determines a "restriction" relation, based on the labels mentioned in the previous paragraph, that is used to determine whether communication is possible and who must initiate communication. Thus the Wiemer-Murray Policy has had the effect of specializing the base DGSA policy of "accomplish it in whatever way you choose, that does not violate other DGSA concepts." [6]

DGSA does not specify the detailed process of transfer of information between domains either. Wiemer indicates that his transfer policy requires specialization of the DGSA security policy to "define both the authorities to transfer information and the rules surrounding the transfer." This specialization would also become part of Wiemer's DGSA policy.

### 5.3 Enterprise Viewpoint

We now look at the example system from each of the RM-ODP viewpoints, starting with the enterprise viewpoint. Because the enterprise viewpoint is constructed as a community of objects, we must specify the communities, which objects are in each community, and the roles of those objects.

#### 5.3.1 Communities

In organizing the system of systems (SoS) for this example, we need to turn first to the communities summarized and represented in Figure 18 on page 54. We have three exclusive communities representing N1, N2, and N3; and, in addition, common communities N1N2, N1N3, N2N3, and N1N2N3. N1 has subcommunities I1, O1, L1, and C1. The same holds true for N2 and N3.

To develop security policy, we must further refine the communities based on the classification of the data that the subcommunities must deal with. Figure 19 on page 55 shows that N1, N2, and N3 all have information objects at LOW classification and HIGH classification.

- Assume that subcommunities I1L, O1, L1, and C1 of N1 share information under the same policy and that N1 gives all its principals permission to use the LOW objects. I1 also has a subcommunity I1H whose principals have permission to use the HIGH classification objects. Assume that the same holds true for N2 and N3.

---

6 Wiemer does make the assumption that all parties use the same structure (with respect to naming) for classification and caveat, i.e., the same categorization is used with the same or related names. This might not be true in policies implemented by commercial parties.

- Assume further that I12, O12, L12, and C12 of N1N2 have objects at LOW classification. I13, O13, L13, and C13 of N1N3 have objects at LOW classification. I23, O23, L23, and C23 of N2N3 have objects at LOW classification. All who are principals of N1 or N2 have permission to use the LOW objects of the N1N2 community. All who are principals of N1 or N3 have permission to use the LOW objects of the N1N3 community. All who are principals of N2 or N3 have permission to use the LOW objects of the N2N3 community.
- Finally, assume that I123, O123, L123, and C123 of N1N2N3 have objects at LOW classification. All who are principals of N1, N2, or N3 have permission to use the LOW objects of the N1N2N3 community.

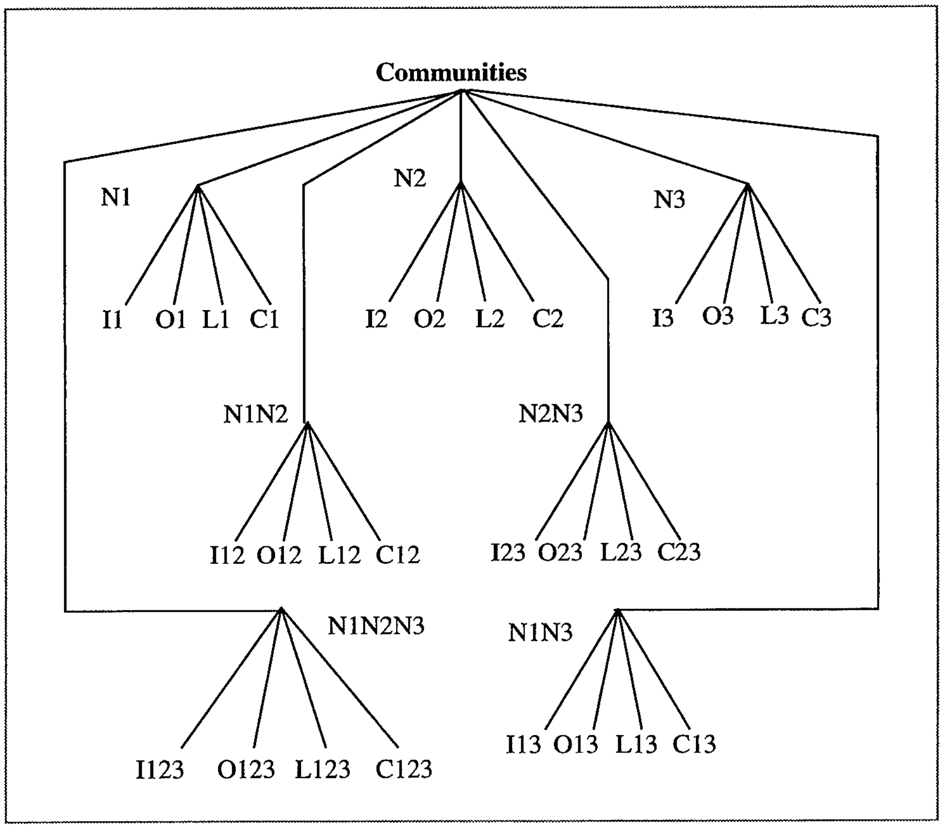



**Figure 18. Communities and Subcommunities**

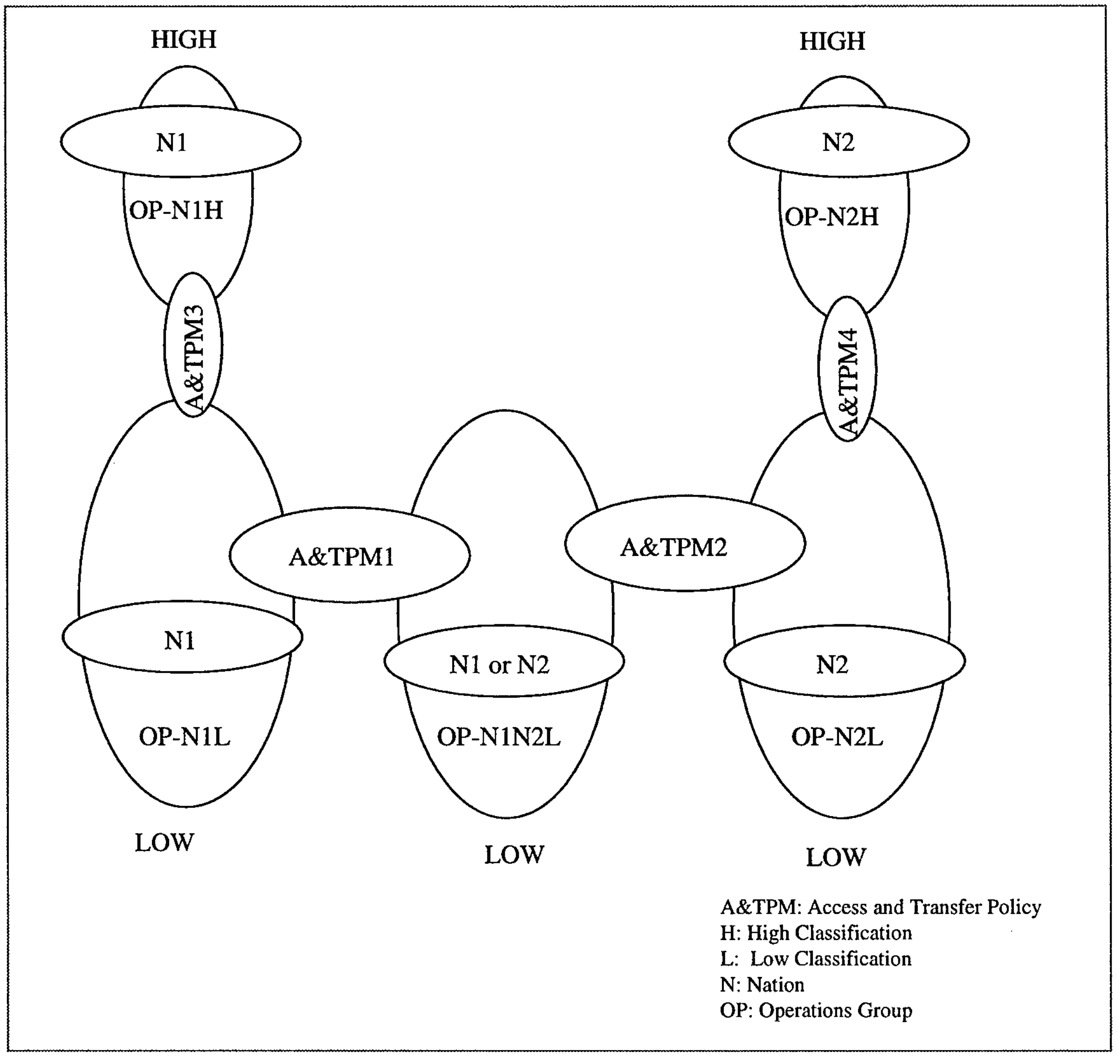



**Figure 19. A Portion of the Wiemer-Murray Coalition Diagram**

### 5.3.2 DGSA Mapping of Information Pools to Information Domains

With each of the ten information pools we associate one information domain: DN1L, DN2L, DN3L, DN1H, DN2H, DN3H, DN1N2L, DN1N3L, DN2N3L, and DN1N2N3L. Principals using the corresponding information pools are users in the corresponding information domains. The "A&TPMi" stands for the combined policies regulating initiation of communication and transfer of information from one information pool to another. The transfer from one information pool to another corresponds to the associated interdomain transfer.

Table 9 on page 56 shows the minimum number of user categories required to populate the example. It is important to note that A,B,C, and D may all be the same principal, or some may

be the same and others distinct. The same is true of the sets of principals for the other nations—that there is no export from DN1N2 to DN3, DN2N3, DN1N3, or DN1N2N3.

In the same manner, there is no export from DN1N3 to DN2, DN2N3, DN1N2, or DN1N2N3, nor from DN2N3 to DN1, DN1N3, DN1N2, or DN1N2N3. That is, there is no direct information flow from a bi-lateral sharing domain to a third party (but there may be indirect flow).

**Table 9. Users and Interdomain Information Flow**

| Premise: (1) at least one principal is a user in both Information Domains and (2) has **Release Authority::Yes** from either domain or **Communication Initiation Rights::Yes** from Domain 1 to Domain 2. | | | |
|---|---|---|---|
| **Principal/User** | **Nationality** | **Information Domain 1** | **Information Domain 2** |
| A | N1 | DN1H | DN1L |
| B | N1 | DN1L | DN1N2L |
| C | N1 | DN1L | DN1N3L |
| D | N1 | DN1L | DN1N2N3L |
| J | N2 | DN2H | DN2L |
| K | N2 | DN2L | DN1N2L |
| L | N2 | DN2L | DN2N3L |
| M | N2 | DN2L | DN1N2N3L |
| V | N3 | DN3H | DN3L |
| W | N3 | DN3L | DN1N3 |
| X | N3 | DN3L | DN2N3 |
| Y | N3 | DN3L | DN1N2N3 |

### 5.3.3 Mapping Principals to DGSA Users

The Wiemer-Murray policy establishes memberships in its security domains through the use of global constraints. These constraints on the sets of DGSA security policies are an attempt to achieve a policy for the coalition community comparable to the operational policy in actual use in coalition warfare.

When we apply these constraints to our example, we find the following to be true (also depicted in Figure 20 on page 58).

- All nationals of N1 who hold the proper level of clearance and need to know are assumed to be able to use information in N1L. They are thus users of DN1L. Usually a subset of N1's nationals have a level of clearance appropriate for HIGH clearance material, and a subset of those have the need to know the intelligence material in I1H (N1H). They are users of DN1H. Because the principals of N1H have a high clearance, they are also principals of N1L.

- In the same way, all nationals of N2 who hold the proper level of clearance and need to know are assumed to be able to use information in N2L. They are thus users of DN2L. Usually a subset of N2's nationals have a level of clearance appropriate for HIGH clearance material, and a subset of those have the need to know the intelligence material in I2H. They are users of DN2H. Because the principals of N2H have a high clearance, they are also principals of N2L.

- Further, all nationals of N3 who hold the proper level of clearance and need to know are assumed to be able to use information in N3L. They are thus users of DN3L. Usually a subset of N3's nationals have a level of clearance appropriate for HIGH clearance material, and a subset of those have the need to know the intelligence material in I3H. They are users of DN3H. Because the principals of N3H have a high clearance, they are also principals of N3L.

- Because all principals of N1L and N2L have sufficiently high clearance levels, fall under the caveat, and have the need to know, they are also principals of N1N2L and hence are users of the information domain DN1N2L.

- Because all principals of N1L and N3L have sufficiently high clearance levels, fall under the caveat, and have the need to know, they are also principals of N1N3L and hence are users of the information domain DN1N3L.

- Because all principals of N2L and N3L have sufficiently high clearance levels, fall under the caveat, and have the need to know, they are also principals of N2N3L and hence are users of the information domain DN2N3L.

- Finally because all principals of N1L, N2L, and N3L have sufficiently high clearance levels, fall under the caveat, and have the need to know, they are also principals of N1N2N3L and hence are users of the information domain DN1N2N3L.

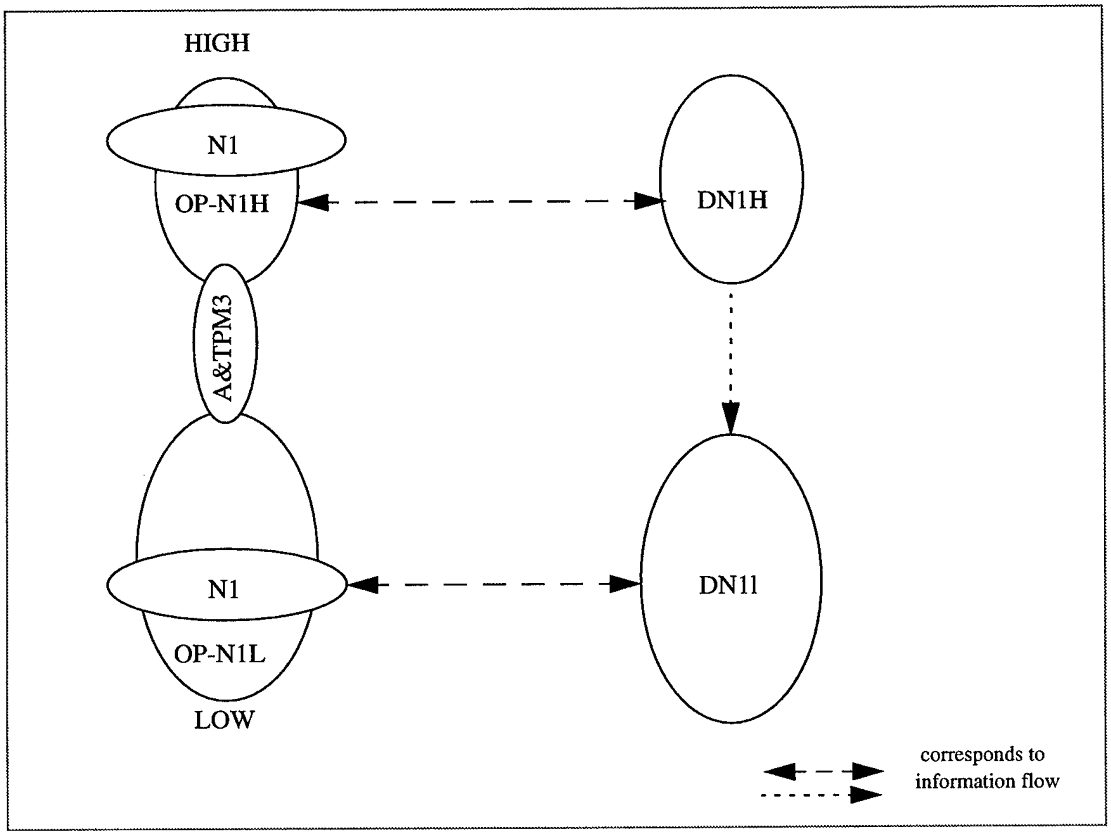



**Figure 20. N1's Information Pools and Domains**

### 5.3.4 Mapping of Roles to DGSA Security Attributes

Each object in the Enterprise viewpoint must have a role. We define here the roles for each of the actors.

- Some principals need to be able to view the plan or access information related to it, but do not need to update it. These users will be said to have assumed the role of *Reader.* They will be assigned a corresponding DGSA security attribute of "Consumer" with the value "Yes," denoted **Consumer**::*Yes*.
- Users who need to be able to generate information as well as access it will have a role of *Producer* and be assigned a corresponding DGSA attribute/value pair of **Producer**::*Yes*.
- Some users may need to execute programs that are represented by information objects. They will be assigned a attribute/value pair of **Executor**::*Yes*. Security policy for each information domain will grant "Read Access," "Write Access," and "Execute Access" when the associated attribute value is Yes.

Note that naming of attributes is relative to the information domain that has the attribute. Because a user has an attribute and value **Producer**::*Yes* in one information domain does not imply that the user has that same attribute or attribute value in a different information domain.

Some principals from each nation need to be able to move information from the nation-specific information pools N1L, N2L, and N3L to the joint pool N1N2L, N1N3L, N2N3L, and N1N2N3L.[7] Because of the security policy we have adopted, these principals need to have the role of *Release Authority*. This role can be reflected in the corresponding DGSA security attribute **Release Authority** and its value:

- For most users of the all information domains, **Release Authority**:: *No*.
- For those privileged to do the release, the value would be *Yes*.

In addition, because of the use of the Wiemer-Murray security policy, we need principals who can establish communications between operations. We will say that the role of such a principal[8] is *Security Officer*. In DGSA there is a corresponding security attribute that might be named **Security Officer** and might have values *Yes* and *No*. These attributes would be used as part of the decision process for granting the right to open a channel from domain Dx to domain Dy. Let users B and K (from the conditions cited previously) have the security attribute/value pair **Security Officer**::*Yes*. User B can initiate connections from DN1 to DN1N2, and user K can initiate connections from DN2 to DN1N2, all according to the Wiemer-Murray policy.

These communities may have many other roles that are related to actions taken as part of the processes for generation of the OPLAN, e.g., *Editor, Reviewer, Author, Commander,* etc. If these roles are not relevant to security decisions, they will not have a corresponding DGSA security attribute.

### 5.3.5 Actions and Processes

At least two action templates will be required because of our use of DGSA and our choice of information domains for subcommunities using N1H, N2H, N3H, N1L, N2L, and N3L:

- The first is "connect Nx, Ny, CRole."

---

7 No direct information flow is permitted from N1H, N2H, or N3H information pools to any of the following coalition information pools: N1N2L, N1N3L, N2N3L, and N1N2N3L. Indirect flow must pass through the appropriate LOW level: N1L, N2L, or N3L.

8 A principal may be a person or a program acting on behalf of a person.

- The second is “transfer IO, Nx, Ny, XRole” where Nx and Ny represent information pools, IO is an information object, CRole is a connection role, and XRole is a transfer role.

At least one action template, “transfer IO, Nx, Ny, XRole,” will be required for subcommunities using N1N2L, N1N3L, N2N3L, and N1N2N3L. Other action templates and process templates may be required that do not interact with the meta-security policy.

### 5.4 Information Viewpoint

The security aspect of the information viewpoint for the CTF example corresponds to the security aspect of the enterprise viewpoint in that it has schemata that represents inter-information-pool transfer, relationships, and associations, and it has separate schemata that represent information, relationships, and associations in each of the information pools. We divide the information viewpoint into two layers, each made up of invariant, static, and dynamic schemata.

#### 5.4.1 Meta-Schemata

One layer (denoted “meta-schemata”) deals with the initiation of connection and transfer of information between information pools as well as the representation and processing of multi-domain information objects. This is illustrated in Figure 21.

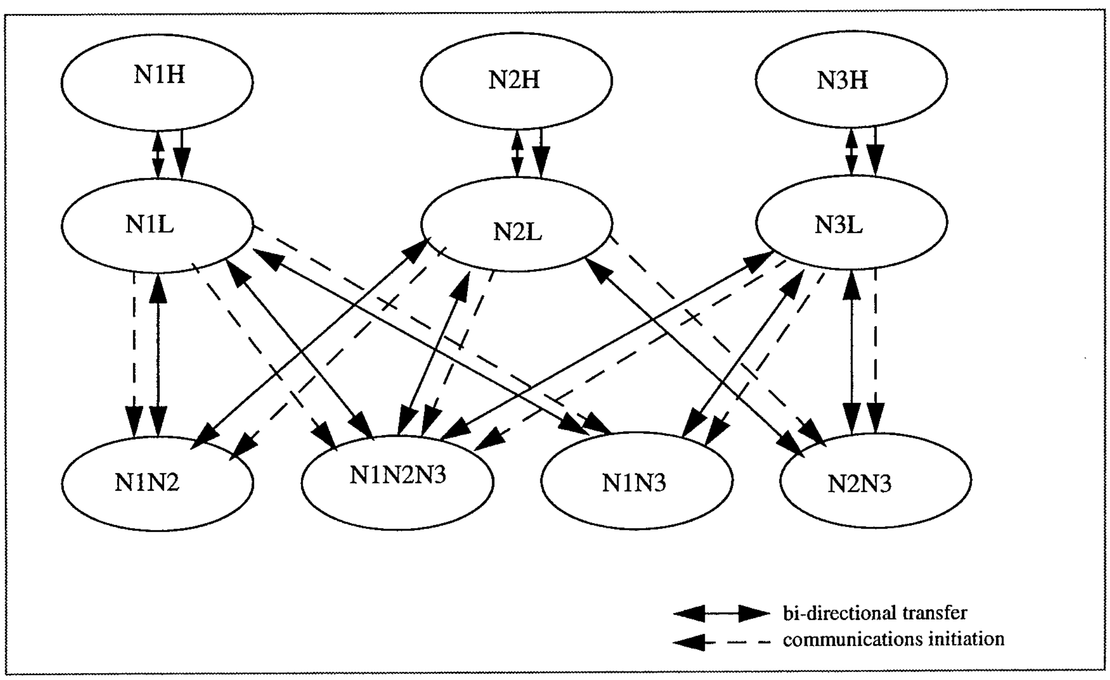



**Figure 21. Meta Schema for the Three Nation Coalition**

Some of the information must be available or computable to answer such questions as the following:

- What other pools can this pool initiate communications to?
- What role(s) is(are) required for doing this?
- What direct information flow is allowed?
- Who controls the flow?
- How are multi-domain objects represented
- What functionality can utilize them?

This set of meta-schemata also describes the information used to represent multi-domain information objects and the functions that may be applied (if any) to them. Examples of such functions might be "print," "display," "move to a new domain," "instantiate," "destroy," "make persistent," etc. Further, any security policy relevant to the representation or use of multi-domain information objects may be partially represented in this set of meta-schema. These schema may also contain information used in managing the security policy, security attributes, and the values of the security attributes contained in the meta-schemata.

In some cases the representation might include an entity for each information pool (information domain) indicating to which information pool information could be exported, whether the pools are "in communication," and the required role or roles of the exporter. Another entity for each domain could indicate to what other pools a principal could initiate communication and the role required for doing this. Alternatively, there might be a set of rules for computing these quantities, for example:

> NxH may initiate communications with NxL for all x, x=1, 2, 3 (information may flow between high and low classification domains of country x if the high classification domain initiates the communication).
>
> NxH may have bi-directional communication with NxL after communications has been initiated by an entity with role Security Officer for all x, x=1, 2, 3.

The DGSA also requires one policy element specific to the information viewpoint. Each information object instance must be uniquely identified, and it must belong to a single information pool in any static schema. In some dynamic schemata, the information object instance may be stripped of its identification and deposited with a new identification in a different information pool

### 5.4.2 IntraDomain Schemata

The other sets of schemata deal with the representation of information and relationships within the information pools. These schemata will contain the security attribute-value pairs for each of the pool-domain mappings. They will also contain representation for each pool of the security policy, Access Decision Function, Access Enforcement Function, and data and functions (relationships and associations) required for the implementation of policy elements relating to integrity and availability. They may also contain information used in managing the security policy, security attributes, and the values of the security attributes contained in the schemata.

Initially, the static schema for each nation will not have an entity with relationships (functions) that permit data transfer to or modifications of other schemata. There will be an entity with a "connect" function. Execution of this function will "cause" a dynamic schema to create entities in the successor static schemata of the exporting party and of the importing party that will permit this data transfer. In addition the dynamic schema will cause a successor static meta-schema to reflect the new connection.

In the CTF example, we could have radically different schemata for the information pools N1L N2L, and N3L:

- The nations may have organizational disparities.
- The nations may have separate subschemata for their intelligence, operations, logistic, and communications operations, and not wish to employ combined schemata to represent their needs.
- The nations may use radically different security policies.

The N1N2N3L, the N1N2L, the N1N3L, or the N2N3L schemata may differ from the individual nation's schema because of the requirement for interoperability between nations.

For our example using N1, N2, and N3, we will assume that the schemata of the three nations are the same as the joint schema and that only the values of the attributes and the number of instances are different.

## 5.5 Computational Viewpoint

In our example, the security aspect of the computational viewpoint corresponds to the security aspect of the enterprise viewpoint and to the security aspect of the information viewpoint in the following manner:

- Actions generally correspond to operations performed on an operation interface.

- Each operation interface is bound to at least one instance of a computational object.

The primary feature that will be evident is that the computational objects will be divided into exclusive groups corresponding to the information pools, and that there will be very limited interaction between computational objects representing different information pools. The form of this interaction consists of "copying and editing an object instance" representing information (or in DGSA terminology an information object) and introducing the (potentially "censored") object instance into a different group of computational objects for use.

Collections of computational objects may be highly dynamic. Initially there are no "connections" between the collections of computational objects belonging to the coalition partners. When the "Connect" operation is invoked by a principal with **Security Officer**::*Yes,* as mentioned previously, an object connecting the information pools is instantiated in the collections belonging to both nations. The new object may then be bound to pre-existing objects in both collections in an appropriate manner by computational objects acting on behalf of principals in those communities. For example, let us call the computational object that performs the copying and editing the "Exporter." The function of this object is to accept objects on behalf of a principal with **Release Authority**::*Yes* and to perform whatever release actions are required, e.g., checking for and eliminating or substituting for specified words; or "fuzzing a picture" in order to avoid passing more information than desired.

What we will see after the "connect A to B" operation has been completed successfully is shown in Figure 22. We have assumed for this example that "connect A to B" enables the connection if issued by the "more restrictive partner," and that this automatically enables transfer from A to B by "Release Authorities" who use the instantiated exporter.

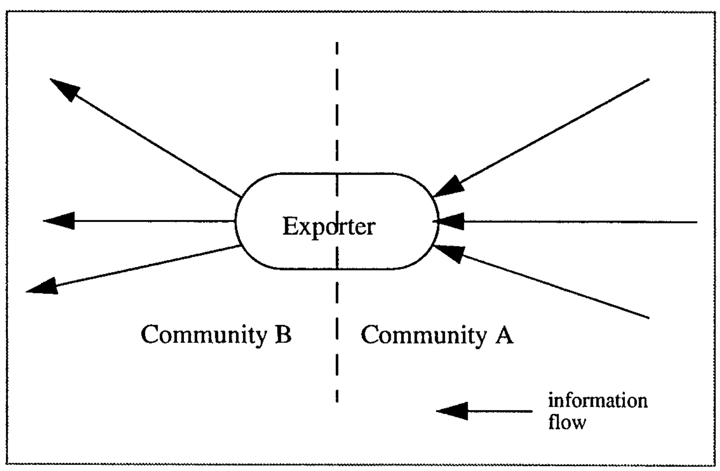



**Figure 22. Partial Computational Model**

Similarly, "connect B to A" would generate an exporter from B to A when issued by the Security Administrator of A, assuming it has been completed successfully. Then members of the B community could export material to the A community. Of course, the principal serving

as security officer in either A or B also has the ability to execute the "destroy self" operation on exporters from his community.

### 5.6 Engineering Viewpoint

The engineering viewpoint of our CTF example has strong correspondences to the computational viewpoint, information viewpoint, and enterprise viewpoint. Usually one or more basic engineering objects are used to represent each information entity or computational object. Additional operation interfaces, signal interfaces, and stream interfaces may be required to implement the relationships represented in the information viewpoint and the functionality represented in the computational viewpoint. Further, management activities may require numerous additional basic engineering objects, clusters, capsules, channels, etc., to facilitate the implementation of dynamic schemata and distributed dynamic features, e.g., transactions. The engineering viewpoint may also contain many additional components required to implement other functionality listed in Chapter 3. This functionality includes location transparency (the network node on which an object resides is not specified in the enterprise, information, or computational viewpoints), as well as the intra- and inter-subcommunity security policies.

The engineering viewpoint must describe the implementation of information domains in enough detail so that assurance arguments can be derived based on stated invariants found in enterprise, information, and computational viewpoints. We note that the DGSA provides a very abstract view of an information domain and the information objects, users, and security policy that make it up. This abstract view permits substantial flexibility in determining an actual implementation. For example, the DGSA does not discuss computing resources except for the use of commercial networks to connect nodes hosting a domain. In particular, it does not discuss how information objects are instantiated, modified, or destroyed, but it does discuss in detail those invariants that must be maintained during their transfer from one node to another or from one domain to another.

It is as if information objects magically appear and are then registered in their domain. There is even a question of whether an information object exists save on a computer screen, in a printed output, or when it becomes a persistent object on computer storage. We assume that part of the reason behind this low level of specificity is that the DGSA is specified in such a way as to minimally constrain the implementation of the security policy chosen for any given information domain. Given the different policies specifying combinations of component policies drawn from the areas of confidentiality, integrity, availability, and accountability, very different implementations may be required.

For the purpose of our example, we will assume that the nucleus maintains separation of all computer-stored persistent information objects in an information domain from those in any other information domain, i.e., the "persistent object stores" are completely separate. Capsules are assumed to separate address spaces of processes in which active objects are being processed. Channels are assumed to separate communication paths traversed by in-transit information objects in different domains while they are routed from one end system to another. It should be noted that channels embody the implementation of security associations from a domain on one end system to another.

We also assume the display system maintains separation of information by domain as displayed by the computer on various display devices, and that physical security prevents unauthorized users from viewing information from domains of which they are not members. Further, the print and backup systems prevent information access by persons who are not members of the appropriate domains by permitting output of representations of information objects to devices that are physically accessible only by authorized users of the domains whose information objects are being portrayed.

All transfers of information from one information domain to another take place on end systems (nodes) hosting the exporting and importing information domains via an intranode channel. Note that the same user is the "owner" of the capsule on each end of the intranode channel, and the same nucleus provides mediation for persistent storage, display, and printing services for both domains.[9] The capsule managers hosting each of the domains coordinate the instantiation of the channel between capsules in accordance with the Wiemer-Murray security policy elements related to connection. Information export between domains is mediated by an interceptor in the channel that enforces the Wiemer-Murray security policy elements related to transfer.

The implementor is required to interpret the DGSA requirement that an information object may only exist on a single node at a time and that all information objects are uniquely identified. This may be done in a number of ways, discussed in the next section, depending on the security policy in force in the domain.

### 5.6.1 Variety of Security Policies

A key concept of the DGSA is that *any* security policy may be employed in an information domain. From an implementation viewpoint, the real questions are:

- How varied are the policies to be used on a single end system?

---

9 This greatly simplifies trust arguments.

- How difficult is it to implement each technical policy on a single end system?
- Is there a generic way in which all potential technical policies for a given enterprise may be supported?

Until one has well-reasoned answers to these questions, it is difficult to propose a structure for the supporting services that will be used on end systems.

For the purposes of this example, we will focus on policies governing "direct" information flows: we will assume that policies restricting "transitive information flow" are not used. In the next section, we will describe a model for access control that meets most of our needs for control of direct information flow although it introduces "systemic inefficiencies" that must be dealt with.

### 5.6.2 The Reading Room Analogy

Users agreeing to share information set up a reading room in which documents containing the shared information may be viewed. Policies for use of the information from those documents are developed and all users must agree to abide by these policies. Only users agreeing to the policy are permitted to use the documents of the reading room.[10] Examples of some of the policies governing reading rooms follow.

### 5.6.3 Document Registration

One method of controlling direct information flow is to model the human handling of classified documents in classified document reading rooms using information objects and DGSA concepts. For the purposes of this analogy, let a document correspond to an information object and the reading room correspond to an information domain. The "librarian" accepts documents that will be kept in the reading room and stamps them with a unique identification number, also including this information in a master journal entry.

### 5.6.4 Copies of Documents

The librarian may make copies of the documents, each with their own unique number and with a reference back to the document that they were copied from, preserving the original identity and suffixing a copy number. Copying a document also requires that information about the copy be entered in a master journal entry. The librarian may loan copies of the docu-

---

10 This analogy is similar to the one put forward by the United Kingdom Ministry of Defence Defense Engineering and Research Agency for acquisition [W98] known as the Domain model. It is simple but may be inefficient if implemented using a single isolated repository. A replicated isolated repository could be chosen to provide greater efficiency—but at additional expense because of the requirement for elaborate consistency protocols.

ments to individuals. The date that the document is loaned and returned, as well as the identity of the reader, is recorded in the journal. If the reader passes the document to another party (who must be a member of the reading room), that event also is recorded as is the return of the document to the first reader.

### 5.6.5 Versions of Documents

At some point, a new version of the document may be generated and cataloged in a given reading room. The librarian may then proceed to destroy any unused copies of the previous version and as each copy of the old document is returned, it may be destroyed if no longer needed. Destruction of a document is recorded. The original is retained.

### 5.6.6 Works Derived from Documents

If the reader produces notes taken from/about a document, the reader is required to inform the librarian who registers the notes. No material may be removed from the reading room except by order of a release authority who causes a copy of the material to be "sanitized" and sent to another reading room where it is logged in with an identification number unique to that reading room. This transfer is recorded in both the sending and receiving reading rooms.

### 5.6.7 Replication of Reading Rooms

Because persons may be located at multiple sites, multiple instances of a reading room may be created in various locations. All instances of the reading room can share the original or copies of documents (depending on policy). Cleared couriers convey the original or copies from reading room to reading room so that copies may be made for users who frequent the rooms. All reading rooms sharing information without requiring export or import operations are in the same information domain; any user of the shared information may access the information at any of the reading rooms housing that information domain.

### 5.6.8 Electronic Implementation of the Reading Room Analogy

We will implement our example based on this reading room analogy. Information objects will be held in a repository, and copies of them will be loaned out to principals in specific domains on specific nodes. The principals may conduct arbitrary calculations based on the information in their address spaces (capsules). Any results that are to persist in nonvolatile storage or to be communicated to different users must be logged into the repository as a new information object. Objects stored in checkpointed engineering objects by means of cluster management are not accessible to users in other domains and are only accessible by the creating user and in the same domain on the same node (and then only to the extent permitted by the security policy).

Another decision that must be made is the degree of distribution that will be permitted to a user's cooperating processes located in separate capsules. In order that the tasks cooperate, channels between the processes must be established over which information may be conveyed. If feasible, point-to-multipoint channels may be used to reduce communication cost and the cost of channel management. On different nodes, if two processes belonging to the same principal and the same domain wish to examine the same information, they may both check out copies of the original information object from the repository and act in parallel. If security policy requires that only one copy is to be used, then the two processes must coordinate and use the single copy in a serial fashion, perhaps by invoking a transaction method.

This leads to the question: Can two or more users of the same domain with identical security attributes and values (other than the principals they represent) collaborate, and if so to what extent? To simplify the example, we will restrict collaboration to the use of the repository. A principal checks out a document, performs a set of tasks, checks in the document, and potentially writes results to the repository. Other principals can then access the document and/or results and continue. This model may not be the most efficient because it limits the use of parallelism, but it is not forced to deal with the situation in which various principals do not share the same collection of roles or other security attributes.

#### 5.6.9 Requirements of Security Associations

A process belonging to a user and operating in an information domain on one end system may wish to connect to another process belonging to that user on a different end system that operates in the same domain. For the connection to be established, a security association between the two processes must be set up. The association bridges the implementation of the security policy, security attributes and values, and set of implementing mechanisms on one end system to that of the other. Multiple associations may be required to connect all interconnections of a user's processes.

#### 5.6.10 Distributed Security Context

The distributed security context contains all the resources used simultaneously by a user in a given information domain. For example, it would contain all processes on all end systems and all security associations operating at a given time. Because of the DGSA requirement of "equivalent protection" for the implementation of information domains on each end system and for each security association, there should be no security "weak point" in the security context regardless how dispersed or how numerous its implementations may be.

### 5.6.11 Inter- and Intra-Node Connectivity

The intra-node channel between a user's capsules is often implemented via inter-capsule communication. We will assume that it does not require encryption or integrity transforms; nor does the information require additional authentication because the nucleus on the node is responsible for separation of information objects, is handling all transport, and has access to the authenticated identities of the sender and receiver.

In the case of an inter-node channel between a user's capsules, we will assume that the information does require encryption and/or integrity transforms, and that it may require additional authentication because the nucleus on the first node has no control over separation of information objects from other domains while that information is in transit to the second node.

When information is to be exported, the mediation between domains must include the examination that the roles of the exporter include *Release Authority,* and then execution of whatever process the security policies require for information to be exported from the first domain and imported into the second domain.

### 5.6.12 Correspondence between Channels and CN

Recall that the communication network (CN) described in the DGSA corresponds to parts of or all of the internode channel described in the RM-ODP. There is a difference in the allocation of security functionality: channels consist of stubs, binding objects, protocol objects, and interceptors. We may allocate encryption, integrity, and authentication functionality to these objects. None of this functionality is assumed to be in the CN. The only allocation of functionality to the CN is availability.

### 5.6.13 Other Reference Models

In some cases the RM-ODP does not offer the proper framework for discussing concerns. An example occurs when considering inter-node channels. Often it is necessary to describe the engineering and technology viewpoints for the communications that occur on the channels. In this case it may be desirable to use supplementary reference models such as the Open System Interconnection Reference Model. This allows the discussion of the media used between systems and its security properties, e.g., optical fiber versus KU-band microwave. It also may allow more detailed discussion of protocol layers not normally dealt with in RM-ODP, e.g., IPSEC.

### 5.6.14 Storage of Information Objects

The RM-ODP does not deal explicitly with the storage of persistent objects except via cluster management and checkpoint/restore functionality. Presumably, these objects can only

be saved/restored through the invocation of a nucleus function by the same principal in the same community. A storage function for data associated with objects could be designed to use services of the nucleus for cataloging, storing, and retrieving data. If this is done, the cataloged data must be protected according to the security policy of each separate domain.

### 5.6.15 Storage and Use of Security Policy

One set of questions must be answered: How are security policies stored? How are they used at runtime? How are they administered?

Policies are stored in Security Management Information Blocks (SMIBs). These are information and therefore are stored in a domain, either the one for which they describe the policy or alternatively in a separate domain specifically for management information. The DGSA document maintains that the choice depends on whether all principals having a special security attribute may manipulate them or not. Because all access of information objects is the same for principals possessing the same security attribute/value pairs, anyone possessing the correct set of pairs could perform maintenance on all objects representing security policy in the information domain if the SMIB is in the domain itself.

For the example, we choose to have the SMIB in a separate domain to which the principal executing the nucleus has read access. When the domain is instantiated for a given user, the nucleus provides the policy to the capsule manager and its enforcement infrastructure. If the policy or user list is changed, the nucleus notifies all capsule managers of that domain that they must refresh their policies and enforcement mechanisms.

### 5.6.16 Multidomain Objects

Another set of questions that must be answered deals with the representation and manipulation of multi-domain objects. For the example, we choose a representation that features a list of opaque pointers in two-tuples.

- The first element of the tuple is a representation identifying the domain that the object is in. This representation is known to all the nuclei of nodes hosting the domain.
- The second element of the tuple is a representation of the unique identification of the object within that domain. Recall that a multidomain object is primarily used for sequencing, printing, or display.

To use the portion of an object within a domain, the two-tuple describing the desired object is exported to the relevant domain with a script that requests an action on that element. The agent receiving the script takes on the delegated identity provided by the sender of the script

and attempts to execute the script under the policy of the domain as it applies to the delegate. It may be that the script will finish and (if the policy permits) return a value indicating success or failure. The policy of the domain may not recognize the delegate as one capable of doing the requested operation(s). It may not permit the return of a value indicating success or failure or of returning an indication where the script failed.

Assuming that the policy permits, the computation in information domain may cause one or more of the following:

- Export of a result to a different information domain according to the policy of the exporting and importing information domains.
- Display of a result on a display external to the system according to the display policy of the information domain.
- Transfer of a result to a printer according to the printing policy of the information domain.

Transporting the information to a representation outside the computer is equivalent to exporting the information to an external domain. Usually the rules for performing such an export insist that a visual representation of the label of the information domain furnishing the information be visually associated with the output. So, for example, paragraphs from a multidomain object will be labeled on the printed material with the "sensitivity" or domain name (category) from whence the information came. In the same manner, they will be labeled on the display screen. Furthermore, on the display screen, elements cannot be copied into regions that do not bear the label of the region from which they are taken.

### 5.7 Technology Viewpoint

The technology viewpoint will reflect the engineering viewpoint most strongly. If different mechanisms are used to provide security services in different capsules representing different information domains, patterns representing different communities, schemata, and groupings from the computational model may be seen. In the same way, if different allocations of security services are used on different nodes in support of information services because they are on heterogeneous equipment, different patterns will be seen. Assuming homogeneous nodes and allocation of services, it may be that nothing in the technology viewpoint will show that information domains are being employed nor their variety.

### 5.8 Achieving the Coalition Objective

The following is a scenario for the achievement of the coalition objective of preparing an OPLAN.

#### 5.8.1 Orders to Prepare the OPLAN

The commander of the CTF sends out an all-forces e-mail in the N1N2N3 domain, indicating the general parameters for the preparation of the OPLAN. Each person who has a part in the preparation begins preparation in his or her unit's primary domain. That is, someone preparing communication coordination from Nation 2 would work in N2L. Someone preparing communication coordination between Nation 2 and Nation 3 would work in N23. Someone preparing intelligence information in Nation 1 would work in N1H. When intelligence information is needed in N1L, a sanitized version would be released to N1L from N1H.

As output is prepared in each nation that is required by all nations, it is sanitized and released to N1N2N3.

#### 5.8.2 Distribution of Work

Work is done by those having participant status, and the product of that work perhaps reviewed by those having observer status. For the sake of this simple example, principals of all domains are assumed not to maliciously modify work of others with whom they share information pools.

#### 5.8.3 Initial Rollup

An initial rollup of each nation's plan is done in that nation's LOW information pool. An intelligence addendum is done in each nation's HIGH information pool. An initial rollup of joint plans is done in N1N2, N1N3, N2N3, and N1N2N3.

#### 5.8.4 Finalization

The national commanders assess the work of their own nations and issue instructions for finalization. They assess the joint operations in the joint information pools and confer on how joint plans should be changed for finalization.

#### 5.8.5 The Final Document

The final document will consist of a multidomain object located, for example, in N1N2N3 for the convenience of all. It will consist of the following:

- Object pointers to the "Three Nation OPLAN" that will be in N1N2N3;
- Each of the "Two Nation Appendices" in N1N2, N1N3, N2N3;

- Each single nation instruction in NiL (i=1, 2, 3); and
- Each intelligence appendix in NiH(i=1, 2, 3).

### 5.8.6 Distribution

The multidomain object is distributed to all interested parties as permitted by policy, who may choose to print parts to which they are entitled by policy. For example, someone with a HIGH clearance from N3 could see intelligence appendices from Nation 3, instructions for Nation 3, joint plans for N1N3 and N2N3, and the main body of N1N2N3 plans. They would not be able to print any other appendix because the policy of the domains would reject the attempt of the principal to print objects in domains where they are not members.

### 5.8.7 The OPORD

Using the OPPLAN, members of each nation prepare the OPORD with respect to their nation in pools of either LOW or HIGH classification. Where necessary, orders affecting multiple nations are prepared using two- or three-nation pools. Distribution is accomplished in a manner similar to that in Section 5.8.6.

In closing, we note that this example has been grossly simplified to avoid obscuring the use of information domains in this example. It is likely that more subcommunities would exist and that they would not wish to share as widely as has been suggested, following the principle that the fewer members of a domain the better (in analogy to *least privilege*).

# 6. Future Activity

Security policies are best expressed in terms understood by the enterprise that issues them, but must be implemented using the available technologies. Since there is no clear mapping from one to the other, the security provided does not match what is desired. Often technology solutions are chosen with little understanding of how they will contribute to the desired security. This can lead to a waste of money along with security weaknesses. We have shown that the RM-ODP separates the concerns in the enterprise viewpoint from those in the technology viewpoint, while providing the information, computational, and engineering viewpoints and reference points between them to bridge the gap.

One of the biggest problems with the transition from enterprise viewpoint to technology viewpoint is that most commercial systems available today provide little support for isolation of information domains as defined in the DGSA. One possible mechanism is virtual machines that provide the functionality of the engineering viewpoint capsules described in this paper. Ideally, a virtual machine would be able to run many of the commercial and legacy applications on which the DoD relies. These virtual machines would run on a kernel that provides a capsule manager for each capsule and the channels used to communicate between capsules, as described in Section 3.4 on page 25. When a channel crosses platform boundaries, a virtual private network is needed to support a security association. Various commercial virtual machine and virtual private network products should be examined for their suitability and to suggest enhancements.

Rather than continually map multiple vocabularies, we might attempt to transform the DGSA specification into a specification using only the RM-ODP viewpoint languages, giving additional examples of what constraints must be added to produce real systems that are DGSA compliant. This would be especially valuable if it could lead us to a generic structure that accepts a policy written in the enterprise language and produces (1) the appropriate schemata in the information viewpoint, (2) interfaces, objects, and bindings in the computational viewpoint, (3) the automatic allocation of security services in the engineering viewpoint.

## References


[Bell-LaPadula] David E. Bell and Leonard J. LaPadula. 1974. *Secure Computer Systems: Mathematical Foundations and Model.* M74-244. MITRE Corporation, Bedford MA.

[BS1998] Gordon Blair and Jean-Bernard Stefani. 1998. *Open Distributed Processing and Multimedia.* Addison Wesley, Reading MA. ISBN 0-201-17794-3

[CC] *Information Technology—Security Techniques—Evaluation Criteria for IT Security.* International Standard ISO/IEC 15408, December 1998. http://csrc.nist.gov/cc/ccv20/ccv2list.htm.

[CJCS1997] Office of the Chairman of the Joint Chiefs of Staff. August 22, 1997. *Defensive Information Operations Implementation.* Document CJCSI 6510.01B.

[ClarkWilson] David D. Clark and David R. Wilson. 1987. A comparison of commercial and military computer security policies. In *Proceedings IEEE Symposium on Security and Privacy*, Oakland, CA, 184-194.

[DGSA] Defense Information Systems Agency, Center for Standards. 1996. *Department of Defense (DoD) Goal Security Architecture (DGSA), Version 3.0.* Volume 6 of [TAFIM]. http://www-library.itsi.disa.mil/tafim/tafim3.0/pages/tafim_6.htm.

[DTOS] Spencer E. Minear. 1995. Providing policy control over object operations in a Mach based system. In *Proceedings of the Fifth USENIX UNIX Security Symposium*, Salt Lake City, UT, 141-156.

[FM98] Edward A. Feustel and Terry Mayfield. 1998. The DGSA: Unmet information security challenges for operating system designers. *Operating Systems Review 32*,1 (January), 3-22.

[ISO10165-7] *Information Technology—Open Systems Interconnection—Structure of Management Information: General Relationship Model.* International Standard ISO/IEC 10165-7, ISO/IEC, 1996.

[ISO10181-1] *Information Technology—Open Systems Interconnection—Security Frameworks in Open Systems—Part 1: Overview.* International Standard ISO/IEC 10181-1, ISO/IEC, 1996.

[ISO10181-2] *Information Technology—Open Systems Interconnection—Security Frameworks in Open Systems—Part 2: Authentication.* International Standard ISO/IEC 10181-2, ISO/IEC, 1996.

[ISO10181-3] *Information Technology—Open Systems Interconnection—Security Frameworks in Open Systems—Part 3: Access Control.* International Standard ISO/IEC 10181-3, ISO/IEC, 1996.

[ISO10181-4] *Information Technology—Open Systems Interconnection—Security Frameworks in Open Systems—Part 4: Non-Repudiation.* International Standard ISO/IEC 10181-4, ISO/IEC, 1996.

[ISO10181-5] *Information Technology—Open Systems Interconnection—Security Frameworks in Open Systems—Part 5: Confidentiality Framework.* International Standard ISO/IEC 10181-5, ISO/IEC, 1996.

[ISO10181-6] *Information Technology—Open Systems Interconnection—Security Frameworks in Open Systems—Part 6: Integrity Framework.* International Standard ISO/IEC 10181-6, ISO/IEC, 1996.

[ISO10181-7] *Information Technology—Open Systems Interconnection—Security Frameworks in Open Systems—Part 7: Security Audit and Alarms Framework.* International Standard ISO/IEC 10181-7, ISO/IEC, 1996.

[ISO13244] *Information Technology—Open Distributed Management Architecture.* International Standard ISO/IEC WG-13244 (ITU-T X.703), ITU and ISO/IEC, 1997.

[ISO15414] *Information Technology—Open Distributed Processing—Reference Model—Enterprise Viewpoint.* Working draft ISO/IEC 15414 | ITU-T X.911, ITU and ISO/IEC. ISO/IEC JTC1/SC7/WG3. 1999. http://enterprise.Systemhouse.MCI.com/WG7/documents.html.

[JV2010] Office of the Chairman of the Joint Chiefs of Staff. Joint Vision 2010. http://www.dtic.mil/jv2010/jvpub.htm

[Leary1995] Leary, John. 1995. Personal conversation with the authors. Pittsburgh, PA: Software Engineering Institute.

[Mayfield1999] Mayfield, Terry. 1999. Personal conversation with the authors. Alexandria, VA: Institute for Defense Analyses.

[NSA91] National Security Agency. 1991. *Integrity in Automated Information Systems.* National Computer Security Center, Fort George G. Meade, MD.

[NSA93] National Security Agency. 1993. *Department of Defense (DoD) Information Systems Security Policy.* National Security Agency, Fort George G. Meade, MD. DISSP-SP.1.

[PART1] ISO/IEC JTC1/SC21/WG7. 1998. *Open Distributed Processing—Reference Model—Part 1: Overview.* International Standard ITU-T X.901 | ISO/IEC 10746-1, International Telecommunication Union (ITU) and ISO/IEC. ftp://ftp.dstc.edu.au/pub/arch/RM-ODP/PDFdocs/part1.pdf

[PART2] ISO/IEC JTC1/SC21/WG7. 1996. *Open Distributed Processing—Reference Model—Part 2: Foundations.* International Standard ITU-T X.902 | ISO/IEC 10746-2, ITU and ISO/IEC. ftp://ftp.dstc.edu.au/pub/arch/RM-ODP/PDFdocs/part2.is.pdf

[PART3] ISO/IEC JTC1/SC21/WG7. 1996. *Open Distributed Processing—Reference Model—Part 3: Architecture.* International Standard ITU-T X.903 | ISO/IEC 10746-3, ITU and ISO/IEC. ftp://ftp.dstc.edu.au/pub/arch/RM-ODP/PDFdocs/part3.pdf

[PART4] ISO/IEC JTC1/SC21/WG7. 1998. *Open Distributed Processing—Reference Model—Part 4: Architectural Semantics.* International Standard ITU-T X.904 | ISO/IEC 10746-4, ITU and ISO/IEC, Switzerland. ftp://ftp.dstc.edu.au/pub/arch/RM-ODP/PDFdocs/np43.pdf

[SS75] Jerome H. Saltzer and Michael D. Schroeder. 1975. The protection of information in computer systems. *Proceedings of the IEEE 63*,9 (September), 1278-1308. IEEE,

Piscataway, NJ. http://web.mit.edu/Saltzer/www/publications/protection/index.html.

[S99] Edward A. Schneider. 1999. Security architecture-based system design. In *Proceedings of New Security Paradigms Workshop*, Alton, Ontario, Canada. ACM Press. http://atlas.ida.org/nspw99.pdf.

[SFR97] Edward A. Schneider, Edward A. Feustel, and Ronald S. Ross. 1997. *Assessing DoD Goal Security Architecture (DGSA) Support in Commercially Available Operating Systems and Hardware Platforms.* IDA Paper P-3375, Institute for Defense Analyses, Alexandria, VA.

[TAFIM] Defense Information Systems Agency, Center for Standards. 1996. *Department of Defense Technical Architecture Framework for Information Management (TAFIM), Version 3.0*, 8 volumes. http://www-library.itsi.disa.mil/tafim/tafim3.0/pages/tafim.htm.

[W98] Simon Wiseman. 1998. Pack 79. Defence Evaluation and Research Agency, Ministry of Defense, Malvern, U.K. Documents DERA/CIS/CIS3/[TR980265-76] (Various Versions).

[WM98] Douglas J. M. Wiemer. 1998. Wiemer-Murray domain security policy model for international interoperability. In *Proceedings of 21st National Information Systems Security Conference*, Alexandria, VA, 525-536. http://csrc.nist.gov/nissc/1998/proceedings/paperF20.pdf.

# Glossary

**abstract architecture** — 1. "Starts with requirements and defines functions to be performed." 2. "Cites principles, fundamental concepts, and functions that satisfy the typical security requirements." 3. "Concepts and functions are allocated to elements of an abstract definition of information systems architecture." [DGSA 1.3.1[1]]

**access transparency** — "A distribution transparency which masks differences in data representation and invocation mechanisms to enable interworking between objects." [PART3]

**accountability** — System activities can be traced to persons or processes that may then be held responsible.

**action template** — A template specifying a policy for the action and a role required to perform the action. At least one principal with the role of actor is associated with each action. When instantiated, the actor(s) performing the action are selected and given the required role necessary to perform the action. The artifacts and resources required to complete the action are also identified and allocated. After the action has taken place, unused resources are released.

**activity template** — A template specifying an activity policy, roles required to complete the activity, and an ordered set of actions that must be accomplished in order to complete the activity. When instantiated, the actor(s) performing the activity are selected and assume the required role necessary to perform the activity. The artifacts and resources required to complete the activity are also identified and allocated. After the activity has taken place, unused resources are released.

**actor** — A role taken on by a principal that performs an action.

**architecture** — Specifically for this paper, a constraint-based architecture: An abstraction for specifying or describing the way a system is organized or constructed using a prioritized list of constraints to determine an acceptable set of implementations [Leary1995].

Architectures provide a means to think about something before, during, and after building it. They allow the expression of concepts, structural forms, functions, and properties through the use of views reflecting desirability, possibilities, and constraints [Mayfield1999].

A set of rules to define the structure of a system and the interrelationships between its parts. [PART2]

**artifact** — A role taken on by an object that is acted upon, e.g., a principal, a data object, or a resource.

---

1. Numbers are sections of [DGSA].

| | |
|---|---|
| **availability** | Access to data and information services is timely and reliable for authorized users. |
| **binding object** | A binding object is used in the computational viewpoint to connect two interacting objects. |
| **capsule** | A capsule is used in the engineering viewpoint to contain capsule managers, cluster managers, engineering objects and channels. |
| **capsule manager** | A capsule manager is used in the engineering viewpoint to control allocation and deallocation of cluster managers and channels. |
| **channel** | A channel is used in the engineering viewpoint to convey information from a basic engineering object in one capsule to a basic engineering object in another capsule. |
| **cluster manager** | A cluster manager is used in the engineering viewpoint to manage a collection of basic engineering objects including instantiation, deletion, checkpoint, and restore. |
| **Common Object Request Broker Architecture (CORBA)** | CORBA is an architecture specified by the Object Management Group that provides communication between objects and location transparency for referenced objects in a distributed system. |
| **communication network** | A term used in this paper to refer to both dedicated and public wide-area networks. |
| **community** | A basic community has an objective; a contract to reach the objective; a set of roles played in reaching the objective; a policy of the community for reaching the objective; a set of activities used to reach the objective; a set of principals who agree to the contract, are assigned a set of roles in the contract, and accept a set of obligations as necessary to achieve the contract; and a set of resources that it uses in order to reach the objective. |
| **Community Equivalent Object (CEO)** | An enterprise object may itself be represented as a community at a lower level of abstraction. |
| **computational language** | The language used in describing specifications of the computational viewpoint. |
| **computational viewpoint** | The model for the transformation of information of the enterprise. |
| **confidentiality** | Information is not made available or disclosed to unauthorized individuals, entities, or processes. |
| **CONPLAN** | A concept plan that is an operation plan in an abbreviated format that would require considerable expansion or alteration to convert it into an OPLAN or OPORD. A CONPLAN contains the CINC's Strategic Concept and those annexes and appendixes deemed necessary by the combatant commander to complete planning. Generally, detailed support requirements are not calculated and TPFDD files (Time Phased Force and Deployment Data) are not prepared. |
| **CONPLAN with TPFDD** | A CONPLAN with TPFDD (Time Phased Force and Deployment Data) is the same as a CONPLAN except that it requires more detailed planning for phased deployment of forces. |
| **constraint-based architecture** | This amounts to a selection (as a partial order) of the constraints in their original priority order. Only those constraints that are relevant to the problem of interest are included and only the detail required by those con- |

straints needs to be examined. Such architectures are not implementation architectures but are *abstractions* of implementation architectures.

A constraint-based architecture is determined by an ordered set of prioritized constraints that reduce the number of implementations of the architecture as more constraints are added to the set. The ideal constraint-based architecture supplies no more constraints than absolutely required for the purpose of the architecture so that the architect/implementer has the greatest flexibility. In the event that the set of constraints permits more than one implementation, the constraint based architecture can be said to be *abstract*.

**consumer** An entity that consumes information.

**Data Encryption Standard** Describes methods of encrypting and decrypting information defined by National Institute for Standards and Technology of the Department of Commerce in a Federal Information Processing Standard (FIPS) no. 46-2

**Department of Defense Goal Security Architecture (DGSA)** The description of an objective (or goal) architecture that should describe the endpoint of the evolution of all DoD security architectures.

**Discretionary Access Control** A element of security policy permitting access to information based on a security attribute of the information, which can be changed by the principal executing a program utilizing the information.

**Distributed Computing Environment (DCE)** A set of software produced by The Open Group providing services that permit the development of distributed applications.

**distribution transparency** "The property of hiding from a particular user the potential behaviour of some parts of a distributed system." [PART2]

**dynamic schema** A dynamic schema consists of two static schema. It corresponds to a snapshot of a system at the beginning of an epoch and at the end of that epoch. It may also dictate one or more possible sets of state transitions by which the final schema is achieved from the beginning schema.

**engineering language** The language used in describing specifications of the engineering viewpoint.

**engineering viewpoint** The model supporting the computational viewpoint that provides "distribution transparencies," including access, failure, location, migration, relocation, replication, persistence, and transaction transparencies.

**enterprise language** The language used in describing specifications of the enterprise viewpoint.

**enterprise viewpoint** The purpose for the system, the use of the system, and the policies for the use of and management of the system. In the RM-ODP, organizational security policy is described in the enterprise viewpoint.

**equivalent domains** Two information domains that share exactly the same set of characteristic labels.

**failure transparency** "A distribution transparency which masks, from an object, the failure and possible recovery of other objects (or itself), to enable fault tolerance." [PART3]

**generic architecture** "Proceeds from an initial allocation of security services and functions and begins to define the types of components and security mechanisms that

| | |
|---|---|
| | are available to implement the security services with particular strength." [DGSA 1.3.2] |
| **information domain** | The elements of an Information Domain are information objects (artifact objects and resources), users (actor objects), and a security policy (obligations, permissions, and prohibitions for each user). |
| **information pool** | Shared collection of information used by a community to achieve an objective. Each information pool is governed by a single security policy. All information in the pool is treated in the same manner. |
| **information language** | The language used in describing specifications of the information viewpoint. |
| **information viewpoint** | The information entities, their relationships, and their associations that are represented and manipulated in the operation of the enterprise. |
| **integrity** | Code or data is not altered or destroyed in an unauthorized manner. |
| **invariant schema** | A schema of the information viewpoint that does not change for the enterprise as a function of time or security domain. |
| **less restrictive domain** | One of two information domains in which the protective mechanisms associated with the characteristic labels of that domain imply lesser protective mechanisms than of the other, and the domains are not *mutually exclusive*. |
| **local subscriber environment** | A safe, protected environment in which DoD end systems can reside (i.e., an enclave). |
| **location transparency** | "A distribution transparency which masks the use of information about location in space when identifying and binding to interfaces." [PART3] |
| **logical architecture** | "Subjects a generic architecture to hypothetical requirements resulting in a detailed example (no cost analysis required)." [DGSA1.3.3] |
| **Mandatory Access Control** | A element of security policy permitting access to information based on a security attribute of the information, which can be changed by the principal executing a program utilizing the information. |
| **meta-schema** | A schema that is used to describe the transfer of information from one information pool to another. |
| **migration transparency** | "A distribution transparency which masks, from an object, the ability of a system to change the location of that object." [PART3] |
| **more restrictive domain** | One of two information domains in which the protective mechanisms associated with the characteristic labels of that domain imply greater protective mechanisms and the domains are not *mutually exclusive*. |
| **multidomain object** | Meta-objects that are compositions of information objects from different information domains and are used to display and transport conglomerations of information. |
| **MultiLevel Secure** | Denotes a node that maintains information in multiple pools, each of which may have different levels of sensitivity. |
| **mutually exclusive domains** | Two information domains that are neither equivalent, more restrictive, or less restrictive (i.e., NATO and AUSCANNZUKUS are mutually exclusive information domains). |
| **nucleus** | The collection of software that manages a node. |

| | |
|---|---|
| **operations order (DoD)** | A directive issued by a commander to subordinate commanders for the purpose of effecting the coordinated execution of an operation. Also called OPORD. |
| **operations plan (DoD)** | Any plan, except for the Single Integrated Operation Plan, for the conduct of military operations. Plans are prepared by combatant commanders in response to requirements established by the Chairman of the Joint Chiefs of Staff and by commanders of subordinate commands in response to requirements tasked by the establishing unified commander. Operation plans are prepared in either a complete format (OPLAN) or as a concept plan (CONPLAN). The CONPLAN can be published with or without a Time Phased Force and Deployment Data (TPFDD) file. |
| **OPLAN** | An operation plan for the conduct of joint operations that can be used as a basis for development of an operation order (OPORD). An OPLAN identifies the forces and supplies required to execute the CINC's Strategic Concept and a movement schedule of these resources to the theater of operations. The forces and supplies are identified in TPFDD files (Time Phased Force and Deployment Data). OPLANs will include all phases of the tasked operation. The plan is prepared with the appropriate annexes, appendixes, and TPFDD files as described in the Joint Operation Planning and Execution System (JOPES) manuals containing planning policies, procedures, and formats. |
| **persistence transparency** | "A distribution transparency which masks, from an object, the deactivation and reactivation of other objects (or itself)." [PART3] |
| **policy** | "A set of rules related to a particular purpose. A rule can be expressed as an obligation, a permission or prohibition." [PART2] |
| **policy invariants** | Features of a policy that do not change in different epochs. |
| **principal** | A principal may be a person or a program acting on behalf of a person. |
| **process** | A process is a set of interrelated activities, possibly performed in different communities. |
| **producer** | A provider of information. |
| **protocol object** | An engineering object part of a channel required to maintain end-to-end channel integrity. |
| **reader** | A role permitting the actor in the role to read information. |
| **reference models** | An agreed language and semantics for expressing a set of related concerns and a commonly assumed model based on a common taxonomy (abstraction) of those concerns. In the standards community, these abstractions are called reference models. |
| **release authority** | A person or program who can approve the transfer of information from one security management domain to another. |
| **relocation transparency** | "A distribution transparency which masks relocation of an interface from other interfaces bound to it." [PART3] |
| **replication transparency** | "A distribution transparency which masks the use of a group of mutually behaviorally compatible objects to support an interface." [PART3] |
| **Security Management Information Base (SMIB)** | The DGSA specifies that the security policy for an Information Domain be specified as a set of objects, known as a Security Management Information Base (SMIB). |

| | |
|---|---|
| **Security Officer** | A person whose role is to manage security functionality associated with a system of systems. |
| **sited domain** | That portion of an information domain that is located on a particular network node. |
| **specific architecture** | "Uses real requirements to permit acquisition or development by component"; "addresses components, interfaces, standards, performance, and cost." [DGSA 1.3.4] |
| **static schema** | A static schema consists of a set of entities, their relationships, and their associations. It corresponds to a snapshot of a system at a given time. |
| **stub object** | A basic engineering object in the engineering viewpoint whose function is to marshall arguments of a remote procedure call, converting the representation of each argument to a canonical form. |
| **system of systems** | A system that is a composition of systems. |
| **technology language** | A language used to describe specifications of the technology viewpoint. |
| **technology viewpoint** | The model describing the components and standards used in the engineering viewpoint. |
| **transaction transparency** | "A distribution transparency which masks coordination of activities among a configuration of objects to achieve consistency." [PART3] |
| **transfer system** | With regards to information transfer, that portion of the enclaves at both ends of a network, relay systems within a network, and the network itself, and their various combinations. |

## Acronyms

| | |
|---|---|
| A&TPM | Access and Transfer Policy |
| API | application programming interface |
| APM | Access Policy Mechanism |
| AUSCANNZUKUS | Australia, Canada, New Zealand, United Kingdom, United States |
| B | binder; binding object |
| C | control object; caveat label; communications |
| CAPI | Cryptographic Application Programming Interface (Microsoft) |
| CEO | Community Equivalent Object |
| CG | command group |
| CINC | Commander in Chief |
| CL | classification label |
| CN | communication network |
| CORBA | Common Object Request Broker Architecture |
| CPM | capsule manager |
| CLM | cluster manager |
| CONPLAN | concept plan |
| COTS | commercial off-the-shelf (software) |
| CSDA | Common Data Security Architecture (Intel) |
| CTF | Coalition Task Force |
| DCE | Distributed Computing Environment |
| DES | Data Encryption Standard |
| DGSA | Department of Defense Goal Security Architecture |
| DLL | Dynamic Link Libraries |
| DoD | Department of Defense |
| ES | end system |
| FIPS | Federal Information Processing Standard |
| GOTS | government off-the-shelf (software) |
| GSS | General Security Service |
| I | interceptor |
| I/O | input/output |
| IEC | International Electrotechnical Committee |
| IEEE | Institute of Electronic and Electrical Engineers |
| IETF | Internet Engineering Task Force |
| IPSEC | Internet Protocol Security Standard |
| ISO | International Organization for Standards |
| ITU | International Telecommunications Union |
| IXIT | Implementation eXtra Information for Testing |
| L | logistics |
| LCS | local communications system |
| LSE | local subscriber environment |
| MOD | Ministry of Defence |
| N | nation |
| NATO | North Atlantic Treaty Organization |
| O | object; operations |
| Oi | object i |
| ODP | Open Distributed Processing |
| OMG | Object Management Group |
| OP | operational label |
| OPORD | operations order |
| OPLAN | operations plan |
| ORB | Object Request Broker |

| | |
|---|---|
| OSI | Open System Interconnection |
| P | protocol |
| QoP | Quality of Protection |
| QoS | Quality of Service |
| RM | Reference Model |
| RM-ODP | Reference Model for Open Distributed Processing |
| RPC | Remote Procedure Call |
| RS | relay system |
| S | stub |
| SMIB | Security Management Information Base |
| SoS | system of systems |
| TAFIM | Technical Architecture Framework for Information Management |
| TCP/IP | Transmission Control Protocol |
| TINA | Telecommunications Intelligent Networking Architecture |
| TOG | The Open Group |
| TPFDD | Time Phased Force and Deployment Data |
| UK | United Kingdom |
| US | United States |

| REPORT DOCUMENTATION PAGE | *Form Approved*<br>*OMB No. 0704-0188* |
|---|---|



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE<br>November 1999 | 3. REPORT TYPE AND DATES COVERED<br>Final |
|---|---|---|

4. TITLE AND SUBTITLE

Interpretation of the Department of Defense Goal Security Architecture Using the Reference Model for Open Distributed Processing

5. FUNDING NUMBERS

DASW01-98-C-0067
Task order DD-5-1671

6. AUTHOR(S)

Edward A. Schneider, Edward A. Feustel

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER

IDA Paper P-3504

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Security Agency
9800 Savage Rd.
Ft. George G. Meade, MD 20755-6000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, unlimited distribution: 05 July 2000.

12b. DISTRIBUTION CODE

2A

13. ABSTRACT (Maximum 200 words)

U.S. defense information systems and those of its coalition partners are increasingly networked among themselves and to other information systems, all of which are frequently managed by different organizations and subject to different security policies. This is especially true in coalition and joint operations where the networks may be very dynamic. These systems must be capable of protecting multiple changing classifications of information. IDA has previously found the DoD Goal Security Architecture (DGSA) provides a suitable model for expressing the information security of such systems.

This paper uses the Reference Model for Open Distributed Processing, an international standard, to present information system security concepts found in the DGSA. The result is several frameworks that can be instantiated to form an architecture for a particular system. Each framework represents a different concern of the security architect. We illustrate these concepts and concerns using a simple military example of a military operations plan and order for a coalition task force employing forces drawn from three coalition nations. Relevant security requirements from the DGSA are imposed to aid the description of the system. On finishing the paper, the reader should understand the principles of the DGSA; have a rudimentary knowledge of the RM-ODP and understand why it is useful in describing security in distributed systems; and begin to understand how an automated system could be designed that would support coalition operations

14. SUBJECT TERMS

DoD Goal Security Architecture (DGSA), Reference Model for Open Distributed Systems (RM-ODP), security architecture, viewpoints, security policy, information domains, isolation, coalition task forces, coalitions.

15. NUMBER OF PAGES

108

16. PRICE CODE

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UL |